UNIVERSITY OF SOUTHAMPTON

Unmanned Aerial Vehicle Routing and Trajectory Optimisation Problems

by

Walton Pereira Coutinho

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Social, Human and Mathematical Sciences
School of Mathematics

September 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES
SCHOOL OF MATHEMATICS

Doctor of Philosophy

by Walton Pereira Coutinho

In recent years, employing Unmanned Aerial Vehicles (UAV) to collect data and making measurements has gained popularity. Often, the use of UAVs allows for a reduction in costs and improvements of other performance criteria. The academic routing community has acknowledged the interest of companies and organisations in adopting UAVs in their operations. However, constraints due to the flight dynamics of UAVs have often been neglected. Finding feasible trajectories for UAVs in a routing problem is a complex task, but it is necessary to ensure the feasibility of the routes. In this thesis 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 subject to flight dynamics constraints. Motivated by a disaster assessment application, we propose a variant of the UAVRTOP, in which a fleet of autonomous aerial gliders is required to photograph a set of points of interest in the aftermath of a disaster. This problem is referred to as the Glider Routing and Trajectory Optimisation Problem (GRTOP). In this work, we propose a single-phase Mixed-Integer Non-linear Programming (MINLP) formulation for the GRTOP. Our formulation simultaneously optimises routes and the flight trajectories along these routes while the flight dynamics of the gliders are modelled as ordinary differential equations. We avoid dealing with non-convex dynamical constraints by linearising the gliders' Equations of Motion (EOMs), reducing the proposed MINLP into a Mixed-Integer Second-Order Cone Programming (MISOCP) problem. Another contribution of this work consists of proposing a multi-phase MINLP formulation for a modified version of the GRTOP. 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 the 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 STO. We perform computational experiments with real-life instances based on flood risk maps of cities in the UK as well as in a large number of randomly generated instances.

Statement of Authorship

I, Walton Pereira Coutinho, declare that the thesis entitled "Unmanned Aerial Vehicle Routing and Trajectory Optimisation Problems" and the work presented in it are my own contribution. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have quoted from the work of others, the source is always given;
- with the exception of such quotation, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as shown in the List of Publications;

Signed: ..... Date: .....

List of Publications

Parts of this work have been published as:

1. Coutinho, W. P., Battarra, M., Fliege, J., (2018), The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review. Computers & Industrial Engineering, 120, 116-128.

2. Coutinho, W.P., Fliege, J., Battarra, M., (2017), Glider Routing and Trajectory Optimisation Problem in Disaster Assessment. Technical Report. Available at `https://eprints.soton.ac.uk/412745/`.

3. Coutinho, W.P., Fliege, J., Battarra, M., Subramanian, A., (2017), Routing a Fleet of Unmanned Aerial Vehicles: A Trajectory Optimisation-based Matheuristic. Technical Report. Available at `https://eprints.soton.ac.uk/420643/`.

Contents

List of Figures

List of Tables

Acknowledgements

As a man of faith, I would like to express my gratitude to the Lord who has given me all support, love and protection whenever I needed.

I also thank my family, namely, my father Nivaldo, my mother Marluce and my sisters Kylma and Ilka. My special gratitude goes to my beloved Lidiane, who has been my closest friend and best companion, despite the ocean-long distance between us. She has been a blessing to my life by shedding light on my darkest moments and by taking care of me in all aspects. I thank my in-laws (especially Cici and Miriam) for their thoughts and prayers.

I must thank all my relatives for always sending their good wishes and prayers. Fortunately, my family is not limited by blood ties. Therefore, I thank my brothers and sisters in Christ, in Brazil and the UK, who also blessed me through their prayers.

During the course of my life, I have found numerous friends that I consider like brothers. They have been a special part of my life. My special gratitude goes to Allan, Emerson, Jefferson, Anand, Diego, Joyce and Glenn Every-Clayton (my adopted grand parents), Márton, Dori and Sigma Benedek (my Hungarian family), Giulia and Luiz (who always took care of me), Carlos, Simos, Ruth, Stefano, Hattie, Karl, Anne and Ana.

This work would not have been completed without the invaluable help from my supervisor, Prof. Jörg Fliege, my advisor, Dr. Maria Battarra (who not only cared about my work, but also about my well-being), and my examiners, Prof. Daniele Vigo and Dr. Antonio Martinez Sykora. I must also thank Márton, Stefano and Alain for the very interesting (and sometimes completely off-topic) conversations we had at each other's offices. I acknowledge the help from every member of staff I met at the University of Southampton.

Finally, I would like to express my gratitude to everyone who directly or indirectly contributed to this work.

This research has been funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [grant number 202241/2014-9].

"The secret things belong to the Lord our God, but the things revealed belong to us and to our children forever, that we may follow all the words of this law."

Deuteronomy 29:29

To my parents.

Chapter 1

Introduction

The focus of this thesis is on the development of models and algorithms for problems involving routing and trajectory optimisation of unmanned aerial vehicles. In this chapter, we first provide a motivation to the subject of this work. Next, we present this project's objectives and an outline of this thesis.

1.1 Motivation

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 (e.g., an electric or combustion engine). Modern UAVs have been first developed in the 1920s to support military operations in which the presence of human pilots was either impossible or too dangerous (Beard and McLain, 2012; Keane and Carr, 2013). However, UAVs have recently become very popular for logistics and surveillance applications (Tsourdos et al., 2010).

A report from the National Purchase Diary has shown that drones sales have increased 224% in 12 months from April 2015, reaching a total of 200 million dollars (NPD, 2016). Due to their ability of embedding several transmitters, sensors and photographing equipment, UAVs can be used in a large range of applications. Successful cases have been reported in, e.g., aerial reconnaissance (Ruzgiené et al., 2015), aerial forest fire detection (Yuan et al., 2015), target observation (Rysdyk, 2006), traffic monitoring and management (Kanistras et al., 2013), online commerce (Wang et al., 2017), geographic monitoring (Uysal et al., 2015), scientific data collection (Stöcker et al., 2015), meteorological sampling (Elston et al., 2014) and disaster assessment and response (Quaritsch et al., 2010; Xu et al., 2014; Nedjati et al., 2016b). In Hayat et al. (2016), several applications of UAV networks are reviewed. The use of UAVs for 3D mapping is surveyed in

Nex and Remondino (2013). A literature review about the applications of UAVs in humanitarian relief is provided by Bravo and Leiras (2015). More examples of the growing applications of UAVs are presented in Rao et al. (2016).

The academic routing community has acknowledged the interest of companies and organisations in adopting UAVs in their operations. A recent example is the approach of combining UAVs and trucks for distribution activities by dispatching drones from trucks for the last mile distribution within city centres (Ha et al., 2015; Murray and Chu, 2015; Wang et al., 2017). It can been shown that this solution could reduce the truck travel time (and the corresponding CO_2 emissions) of up to 50%. The UAV Task Assignment Problem (UAVTAP) is another name used in the literature for the problem of scheduling and routing a fleet of UAVs (Khamis et al., 2015). A growing body of literature appeared on the UAVTAP in the last decade, see, e.g., Ramirez-Atencia et al. (2016), Wang et al. (2015), Hu et al. (2015a), Thi et al. (2012), Alidaee et al. (2010) and Edison and Shima (2011). However, the UAV routing literature has often neglected constraints due to the flight dynamics of UAVs. Finding feasible trajectories for UAVs in a routing problem is a complex task, but it is necessary to ensure the feasibility of the UAVs routes. 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 of even infeasible for these UAVs.

Most of the UAVs used for civil applications present a low flight autonomy. Therefore, it is important for UAV routing algorithms to properly model battery life. According to Fügenschuh and Müllenstedt (2015), this can be achieved by integrating the UAVs' dynamics with routing. As mentioned by the authors, for powered UAVs, a proper modelling of the actual fuel consumption must include, for instance, the current weight, the altitude, the speed and climb/descend rate, which are usually modelled by flight dynamics.

Zhang et al. (2012) considers a problem where a UAV must visit a set of targets. However, after reaching a predetermined distance from a target the UAV must adjust its flight attitude in order to perform a payload delivery. After the delivery, the UAV must complete an *escape manoeuvre* and prepare for the next delivery. According to Zhang et al. (2012), routing and trajectory optimisation must be integrated in order to ensure the safety of the vehicle and the feasibility of trajectories.

Another example occurs in disaster assessment applications, where 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, evaluate access issues, 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.

The computation of trajectories for UAVs has been widely studied in the aerospace engineering and optimal control literature (Yang et al., 2016). The Trajectory Optimisation (TO) problem consists of finding a control history of a vehicle that minimises a scalar performance index (for example, flight time or fuel consumption) while satisfying constraints on the kinematics (position, velocity and acceleration) and the dynamics (forces and moments) of the vehicle. A trajectory is generally associated to a set of Equations of Motion (EOM) that describe the relationship between spatial and temporal changes of a system. The TO problem is closely related to the Optimal Control (OC) problem (Betts, 2001).

The problem named PP consists of finding a flyable path for a UAV visiting a given sequence of waypoints (targets) in a two-dimensional (2D) or three-dimensional (3D) space without considering the vehicle's dynamics. According to Gasparetto et al. (2015), PP is a *geometric* problem, because it is defined as finding a geometric path regardless any specified time law. In turn, TO consists of assigning a time law to a controlled geometric path.

More complex variants of the PP problem including, for instance, wind and motion constraints, require substantial simplifications and assumptions to be solved heuristically (Kunchev et al., 2006; Rathinam and Sengupta, 2007). The books by Tsourdos et al. (2010) and Beard and McLain (2012) provide good overviews of PP algorithms for UAVs. On the other hand, high fidelity TO models (i.e., using more accurate physical models) have been developed for aircraft and spacecraft (Raivio et al., 1996; Conway, 2010; Fisch, 2011; García-Heras et al., 2014; Colasurdo et al., 2014). These models are currently solved by OC techniques. An overview of OC methods for TO is provided in Betts (1998, 2001).

The field of TO has however not considered routing decisions: given a set of ordered waypoints, it is possible to find a feasible trajectory for a generic UAV, but it is not clear in the literature if the sequence of waypoints is appropriate. For example, for a gliding vehicle (i.e., with no onboard thrust) a given waypoint sequence might be infeasible in terms of flight dynamics. Given a fleet of UAVs, it is an open question how to combine routing and trajectory decisions in a single optimisation problem. As far as the authors are aware, there is not a survey summarising the literature about routing and trajectory optimisation for UAVs.

Research about integrated routing and 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 addition, 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.

In this work, we propose a single-phase Mixed-Integer Non-linear Programming (MINLP) formulation for the GRTOP. Our formulation simultaneously optimises routes and the flight trajectories along these routes while the flight dynamics of the gliders are modelled as ordinary differential equations. We avoid dealing with non-convex dynamical constraints by linearising the gliders' EOMs, reducing the proposed MINLP into a Mixed-Integer Second-Order Cone Programming (MISOCP) problem. Another contribution of this work consists of proposing a multi-phase MINLP formulation for a modified version of the GRTOP. 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.

1.2 Objectives

The objectives of this work are defined as follows.

- Review the literature concerning UAV routing and UAV trajectory optimisation.
- Formally introduce the UAVRTOP.
- Develop a mathematical model for the GRTOP.
- Develop a heuristic method capable of solving large GRTOP instances.

1.3 Thesis outline

- Chapter 2 formally introduces the UAVRTOP and presents a taxonomic review of recent contributions on UAV routing and UAV trajectory optimisation.
- Chapter 3 presents the glider dynamics model that will be used throughout this thesis.
- Chapter 4 presents a single-phase mathematical formulation and exact solutions for the GRTOP.
- Chapter 5 presents a multi-phase mathematical formulation and a heuristic method for solving larger GRTOP instances.
- Chapter 6 concludes this work and provide future research directions.

Chapter 2

Literature Review

The purpose of this chapter is to present the UAV Routing and Trajectory Optimisation Problem (UAVRTOP), highlighting approaches already proposed in the literature and directions for further research. We introduce a taxonomy, that is able to identify the key components of routing and TO problems, as well as highlight assumptions and simplifications commonly adopted in the literature.

2.1 The UAV routing and trajectory optimisation problem

In this section, we formally define the UAV Routing and Trajectory Optimisation Problem (UAVRTOP), the problem in which a fleet of UAVs has to visit a set of waypoints assuming generic kinematics and dynamics constraints. Wind conditions, collision avoidance between UAVs and obstacles can be incorporated in the model as well.

2.1.1 A mathematical formulation for the UAVRTOP

In the following, we assume a fleet C of UAVs is available at the launching site 0. Let $G = (V, A)$ be a graph, where the set V represent all the waypoints that need to be visited by the UAVs and A represent the set of arcs between waypoints. In addition, let $0'$ represent the landing site. The cost of using a vehicle $k \in C$ is F_k. The parameters (e.g., mass, wing area, aerodynamics coefficients) of the UAV k travelling between i and j are stored in the vector $\mathbf{p}_{ijk}$. Note that these parameters may change during the mission due, for example, to a change in the flight mode (if hybrid UAVs are used). The state of a UAV is a vector fully defining the position, orientation and velocity of the vehicle in some coordinate system (alternative state representations will be described in Section 2.2).

For simplicity, we recall $\mathbf{y}_{ijk}(t_{ijk}) \in \mathbb{R}^{n_y^k}$, $n_y^k \in \mathbb{Z}$, the state variable of the UAV k travelling between waypoints i and j at time $t_{ijk} \in \mathbb{R}$. Similarly, the control variables model the inputs that are given to the physical systems in order to achieve a desired trajectory. Typical control variables for UAVs are the thrust (the impulse given by the UAV engine, if any), the roll angle, a.k.a. bank angle (which "banks" the aircraft to change its horizontal flight direction), and the angle-of-attack (which is related to how much lift the aircraft's wing generate). We define $\mathbf{u}_{ijk}(t_{ijk}) \in \mathbb{R}^{n_u^k}$, $n_u^k \in \mathbb{Z}$, the control variables for a UAV k flying on arc (i, j) at time $t_{ijk} \in \mathbb{R}$.

The physical laws governing the UAV k travelling between the waypoints i and j at time t_{ijk} are called the *system dynamics*. In general terms, the system dynamics can be expressed by a set of EOM in the form of a system of Ordinary Differential Equations (ODEs) as follows:

$$\dot{\mathbf{y}}_{ijk} = \mathbf{f}_k(\mathbf{y}_{ijk}(t_{ijk}), \mathbf{u}_{ijk}(t_{ijk}), \mathbf{p}_{ijk}, t_{ijk}) \forall i, j \in V, \forall k \in C \tag{2.1}$$

The functions $\mathbf{f}_k, \forall k \in C$, in the right hand side of the EOM (2.1), represent the relationship between the variables and parameters with the derivatives over time of the state variables (here denoted by " $\dot{}$ ").

State and control variables have to be specified for a time instant to initialise the ODEs. In what follows, we assume that the initial conditions need to be specified at time $t = 0$. It is also reasonable to assume that only the control variables need to be optimised since the values of the states can be determined, provided an initial condition and the evolution of the controls over time.

The routing cost for a UAV k to travel between waypoints i and j can be computed as:

$$\int_{t_{ijk}^o}^{t_{ijk}^f} w_k(\mathbf{y}_{ijk}(t_{ijk}), \mathbf{u}_{ijk}(t_{ijk}), \mathbf{p}_{ijk}, t_{ijk}) \mathrm{d}t_{ijk}. \tag{2.2}$$

The variables t_{ijk}^o and t_{ijk}^f represent the initial and final flight times of the UAV k travelling between waypoints i and j such that $t_{ijk} \in [t_{ijk}^o, t_{ijk}^f]$.

Bounds on the state and control variables are usually imposed by a given UAV technology. We denote $\mathbf{y}_{ijk}^{lb}$ and $\mathbf{y}_{ijk}^{ub}$ the lower and upper bounds on the state variables $\mathbf{y}_{ijk}(t_{ijk})$ of the UAV k travelling on arc (i, j) for all $t_{ijk} \in \mathbb{R}$, respectively. Similarly, $\mathbf{u}_{ijk}^{lb}$ and $\mathbf{u}_{ijk}^{ub}$ represent the lower and upper bounds on the control variables $\mathbf{u}_{ijk}(t_{ijk})$ of the UAV k travelling on arc (i, j) for all $t_{ijk} \in \mathbb{R}$. We also assume lower and upper bounds on the operational constraints, here denoted as $\mathbf{g}_{ijk}^{lb}$ and $\mathbf{g}_{ijk}^{ub}$.

According to our assumption on the initial conditions, the initial flight time from the launching point must be defined as $t_{0jk}^o = 0, \forall j \in V, \forall k \in C$. Let $\bar{\mathbf{y}}_o$ and $\bar{\mathbf{u}}_o$ represent predetermined initial conditions. Thus, the initial state and control variables can be

defined as $\mathbf{y}_{0jk}(t^o_{0jk}) = \bar{\mathbf{y}}_o$ and $\mathbf{u}_{0jk}(t^o_{0jk}) = \bar{\mathbf{u}}_o$, respectively, if UAV k departs from the launching point.

Let us define the following binary variables:

$$x_{ijk} = \begin{cases} 1, & \text{if UAV } k \text{ flies directly from waypoint } i \text{ to } j \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Hereafter, we will describe the optimisation problem defined by Equations (2.4-2.19). This formulation is a conceptual model created for describing the UAVRTOP in mathematical terms. The objective function (2.4) minimises the sum of the fixed cost of using a UAV, the routing cost of flying between waypoints i and j and a measure of the quality of the trajectories at the end points of each arc (i, j). Non desirable features at the end points of the UAVs' trajectories can be penalised in the objective function by means of the functions $\phi_k(\mathbf{y}_{ijk}(t^f_{ijk}), \mathbf{u}_{ijk}(t^f_{ijk}), \mathbf{p}^{ijk}, t^f_{ijk})$. Such undesirable characteristics may include, e.g., sharp flight angles, prohibited flight speeds and noise levels (Vanderbei, 2001; Zhang et al., 2012). Constraints (2.5) and (2.6) ensure that every waypoint is visited exactly once and that, if a UAV arrives at a waypoint $l \in V$, it must also depart from l. Constraints (2.7) make sure that each UAV departs from the launching point 0 and lands in $0'$, if the UAV k is used. Constraints (2.8) ensure that the UAVs' dynamics are preserved if arc (i, j) is used in a solution. In a similar way, Constraints (2.9-2.11) make sure the bounds on the state variables, control variables and *operational constraints* $(\mathbf{g}_{ijk}(\mathbf{y}_{ijk}(t_{ijk}), \mathbf{u}_{ijk}(t_{ijk}), \mathbf{p}_{ijk}, t_{ijk}))$ are respected for every arc (i, j) and for every UAV k if these are travelled in the optimal solution. These constraints can model, for example, obstacles in the space, collisions between UAVs, non desirable flying dynamics, etc. Constraints (2.12) and (2.13) ensure that the final state and control variables at every arc (i, j) visited by UAV k is linked to the state and control variables of its subsequent arc (j, l) if waypoints i, j and l are visited by UAV k in this order. Constraints (2.14) preserve the continuity of the time variable $t_{ijk}, \forall i, j \in V$, along the UAV's k trajectory for all $k \in C$. Constraints (2.15) and (2.16) provide the initial states and controls for every UAV departing from the launching point. Finally, Constraints (2.17-2.19) define the domain of the variables.

The UAVRTOP can be modelled as follows:

$$\begin{aligned} \min \quad & \sum_{k \in C} \sum_{i \in V} F_k x_{0ik} \\ & + \sum_{k \in C} \sum_{(i,j) \in A} \left\{ \int_{t^o_{ijk}}^{t^f_{ijk}} w_k(\mathbf{y}_{ijk}(t_{ijk}), \mathbf{u}_{ijk}(t_{ijk}), \mathbf{p}_{ijk}, t_{ijk}) \mathrm{d}t_{ijk} \right\} x_{ijk} \\ & + \sum_{k \in C} \sum_{(i,j) \in A} \phi_k(\mathbf{y}_{ijk}(t^f_{ijk}), \mathbf{u}_{ijk}(t^f_{ijk}), \mathbf{p}^{ijk}, t^f_{ijk}) x_{ijk} \end{aligned} \tag{2.4}$$

$$\text{s.t.} \quad \sum_{k\in C}\sum_{\substack{i\in V\cup\{0\}\\ i\neq j}} x_{ijk} = 1, \forall j \in V \tag{2.5}$$

$$\sum_{\substack{i\in V\cup\{0\}\\ i\neq j}} x_{ijk} - \sum_{\substack{i\in V\cup\{0'\}\\ i\neq j}} x_{jik} = 0, \forall j \in V, \forall k \in C \tag{2.6}$$

$$\sum_{i\in V} x_{0ik} = \sum_{i\in V} x_{i0'k} \leq 1, \forall k \in C \tag{2.7}$$

$$\dot{\mathbf{y}}_{ijk} = \mathbf{f}_k(\mathbf{y}_{ijk}(t_{ijk}), \mathbf{u}_{ijk}(t_{ijk}), \mathbf{p}_{ijk}, t_{ijk}) x_{ijk}, \forall i, j \in V, \forall k \in C \tag{2.8}$$

$$\mathbf{g}^{lb}_{ijk} x_{ijk} \leq \mathbf{g}_{ijk}(\mathbf{y}_{ijk}(t_{ijk}), \mathbf{u}_{ijk}(t_{ijk}), \mathbf{p}_{ijk}, t_{ijk}) \leq \mathbf{g}^{ub}_{ijk} x_{ijk}, \quad \forall i, j \in V, \forall k \in C \tag{2.9}$$

$$\mathbf{y}^{lb}_{ijk} x_{ijk} \leq \mathbf{y}_{ijk}(t_{ijk}) \leq \mathbf{y}^{ub}_{ijk} x_{ijk}, \forall i, j \in V, \forall k \in C \tag{2.10}$$

$$\mathbf{u}^{lb}_{ijk} x_{ijk} \leq \mathbf{u}_{ijk}(t_{ijk}) \leq \mathbf{u}^{ub}_{ijk} x_{ijk}, \forall i, j \in V, \forall k \in C \tag{2.11}$$

$$\mathbf{y}_{jlk}(t^o_{jlk}) x_{ijk} = \mathbf{y}_{ijk}(t^f_{ijk}) x_{ijk} x_{jlk}, \forall i, j, l \in V, \forall k \in C \tag{2.12}$$

$$\mathbf{u}_{jlk}(t^o_{jlk}) x_{ijk} = \mathbf{u}_{ijk}(t^f_{ijk}) x_{ijk} x_{jlk}, \forall i, j, l \in V, \forall k \in C \tag{2.13}$$

$$t^o_{jlk} x_{ijk} = t^f_{ijk} x_{ijk} x_{jlk}, \forall i, j, l \in V, \forall k \in C \tag{2.14}$$

$$\mathbf{y}_{0jk}(t^o_{0jk}) = \bar{\mathbf{y}}_o x_{0jk}, \forall j \in V, \forall k \in C \tag{2.15}$$

$$\mathbf{u}_{0jk}(t^o_{0jk}) = \bar{\mathbf{u}}_o x_{0jk}, \forall j \in V, \forall k \in C \tag{2.16}$$

$$x_{ijk} \in \{0, 1\}, \forall i, j \in V, \forall k \in C \tag{2.17}$$

$$\mathbf{u}_{ijk}(t_{ijk}) \in \mathbb{R}^{n^k_u}, \forall i, j \in V, \forall k \in C \tag{2.18}$$

$$t_{ijk}, t^o_{ijk}, t^f_{ijk} \in \mathbb{R}. \forall i, j \in V, \forall k \in C \tag{2.19}$$

2.2 The trajectory optimisation problem

Trajectory Optimisation Problems (TOPs) are a special case of OC problems determining the trajectory of a system (e.g., vehicles such as spacecraft, aircraft, UAVs) while minimising a measure of performance and satisfying a set of boundary (initial and final) conditions, path constraints and system dynamics.

The origin of OC problems dates to as early as the 17th century when Johann Bernoulli proposed the Brachistochrone problem (Ross, 2009), one of the first problems in calculus of variations. One of the first applications of the calculus of variations to the control of flying vehicles was presented by Robert Goddard in "A method of reaching extreme altitudes" (Goddard, 1919), where the objective was to determine the minimum initial mass of a ground-based rocket necessary to achieve a given altitude. OC methods are a classical tool in the computation of spacecraft trajectories, e.g., for interplanetary travels and satellite transfer orbits around Earth (Conway, 2010; Colasurdo et al., 2014).

Usually, system dynamics are modelled by a set of EOM that can be nonlinear and discontinuous. *Six degree of freedom* (6DOF) EOM are composed by translational equations (containing forces, position, velocity, acceleration, etc.) and rotational equations

(containing moments, angular velocities, angular acceleration, etc.). Under simplifying assumptions, 6DOF EOM can be decoupled into *three degree of freedom* (3DOF) EOM, see, e.g., Stengel (2004) and Fisch (2011). Usual state variables in 6DOF EOM are the position vector, velocity, pitch angle, pitch rate, weight and flight path angle. In the 3DOF case, the state vector can represent, for instance, the position, velocity, flight path angle and yaw angle of the vehicle.

Solving a TOP for an aircraft consists of generating the inputs for the aircraft's control system so as to perform a desired set of manoeuvres. A TOP takes as input the dynamic constraints of the aircraft and outputs a time indexed sequence states and controls such as positions, velocities and accelerations.

Other difficulties can be added to the problem if one considers that the boundary conditions depend on unknown variables or if the dynamics of the vehicles change over time. In this cases, TOPs can be divided into two or more *phases* in order to properly model the changes in the operational or physical characteristics of the vehicles. A phase can be defined as a segment of a trajectory in which the dynamical system remains unchanged. Phases can be described by their own boundary conditions, system of differential equations, operational constraints and time events. Finally, all phases can be linked or not depending on the behaviour of the dynamical system.

Aircraft TO models have gained much popularity over the last decades. For instance, Schultz and Zagalsky (1972) present solutions for several fixed endpoint aircraft TOPs using calculus of variations. In Raivio et al. (1996), a nonlinear programming-based method is proposed to compute optimal trajectories for a descending aircraft. Fisch (2011) presents a high fidelity optimisation framework for the computation of air race trajectories under safety requirements. García-Heras et al. (2014) compare several OC methods for the TO of cruise flight with fixed arrival time. Finally, Delahaye et al. (2014) present a survey of mathematical models for the computation of aircraft trajectories.

OC methods for UAVs are similar to those of full size aircraft, and therefore similar dynamical systems can be used for both types of planes. On the other hand, new challenges are introduced when specific mission's demands are considered. Moreover, due to their limited capacity, extra effort must be put on determining successful flight plans. Therefore, algorithms that are capable of tackling the UAVs' particularities while developing flight plans must be developed.

2.2.1 Direct and indirect methods for trajectory optimisation problems

Two main classes of numerical methods became very popular for solving TOPs, namely, *direct* and *indirect* methods. The so-called *direct* methods rely on the discretisation of a infinite-dimensional OC problem into a finite-dimensional optimisation problem. This

strategy is commonly known as "discretise, then optimise". In a *direct single shooting* method, for example, the controls are discretised on a fixed grid using an arbitrary parametrisation scheme. The next step of this method consists of solving a non-linear programming problem in order to find an optimal vector of parameters. The *indirect* methods consist of determining necessary optimality conditions for an OC problem and then using a discretisation method to solve the resulting equations. Indirect methods generally apply an "optimise, then discretise" strategy. In an *indirect single shooting* method, for example, the resulting optimality conditions consist of a boundary value problem, which can be solved by means of a simple single shooting algorithm (Betts, 2001).

Several sophisticated algorithms have been developed for solving TOPs. Reviewing such works is considered out of the scope of this work. More information about algorithms for OC and TO can be found, e.g., in the papers by Stryk and Bulirsch (1992), Betts (1998), Ross (2009), Wang (2009) and Rao (2014); and the books by Bryson (1975), Bertsekas (1979), Betts (2001), Bryson (2002) and Kirk (2012).

2.2.2 The UAV path planning problem

Using the notation defined by Latombe (1991), the basic PP problem can be defined as follows. Let $\mathcal{A}$ be an object (a robot) moving in a workspace $\mathcal{S}$ (e.g., in an Euclidean space $\mathcal{S} = \mathbb{R}^n, n = 2$ or 3). A set of obstacles $\mathcal{B}_1, \ldots, \mathcal{B}_m$ is assumed to be distributed over $\mathcal{S}$. The problem consists in, given initial and final *configurations* (position and orientation) for $\mathcal{A}$, find a path in $\mathcal{S}$ that avoids collisions with the objects $\mathcal{B}_1, \ldots, \mathcal{B}_m$. It has been shown that this problem is $\mathcal{NP}$-hard if the velocity of the object $\mathcal{A}$ is unbounded and no rotation is considered (Reif and Sharir, 1994). For Gasparetto et al. (2015), a path planning problem consists of finding a collision-free path among an environment from an initial point to a final goal. For example, YongBo et al. (2017) studies a path planning problem with obstacles in three dimensions. In the literature, the terms PP and *motion planning* are used almost interchangeably (Barraquand and Latombe, 1991). Both problems have gained much popularity over the years. Figure 2.1 shows the number of publications by year on UAV PP problems.

Path planning algorithms can be classified into discrete and continuous methods. In the former, the workspace $\mathcal{S}$ is transformed into a graph through discretisation. Conventional heuristics or exact shortest path algorithms are then used to find a path between given initial and final configurations. The output for discrete methods are usually polygonal paths, i.e., path with no curvature constraints. Therefore, in the case of UAVs these paths need to be further refined. Continuous methods represent $\mathcal{S}$ by using a continuous function. Tsourdos et al. (2010), for instance, employed attraction fields to represent the desired endpoints and repulsive fields to represent obstacles in order to produce a collision-free UAV path.

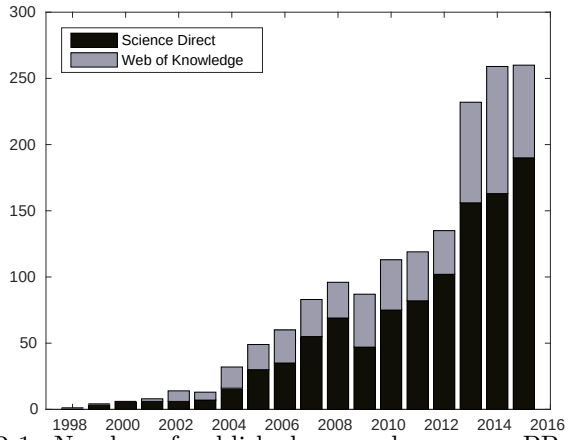


FIGURE 2.1: Number of published papers by year on PP problems.

Problems integrating UAV routing and PP have already been studied before (Manyam et al., 2015; Ho and Ouaknine, 2015; Enright et al., 2015; Sundar and Rathinam, 2014; Levy et al., 2014). Under simplifying assumptions, a PP problem can be modelled as a network problem and standard shortest path techniques can be used. A common assumption is that the UAV can be modelled as a Dubins' vehicle (Medeiros and Urrutia, 2010). A Dubins' vehicle has a limited turning angle and is restricted to move forward, therefore it can be a good representation for some types of UAVs. This simplification is very popular specially for modelling rotary-wing aircraft such as quadcopters. However, for most of the fixed-wing UAVs, the Dubins' assumption might not be suitable due to their complicated dynamics. The reader is referred to Tsourdos et al. (2010) for more details on UAVs PP methods.

Most algorithms for UAV PP have been originated from adaptations of existing algorithms for robot PP. However, we do not intend to survey all PP algorithms as it has already been done in other articles, e.g., Kunchev et al. (2006), Goerzen et al. (2009), Galceran and Carreras (2013) and Yang et al. (2016).

2.3 The vehicle routing problem

The Vehicle Routing Problem (VRP) is a very well known problem in operational research and combinatorial optimisation. In the VRP, routes must be assigned to a set of vehicles that must serve a set of customers such that the total cost of the operation is minimised. Its classical variant is called Capacitated Vehicle Routing Problem (CVRP), where a load capacity is assigned to each vehicle.

The CVRP can be formally defined as follows. A set of vertices $V = \{0, \ldots, n\}$ and a set of arcs A connecting these vertices are given. Each vertex represents a customer with demand d_i, $i \in V \setminus \{0\}$. A value $c_{i,j}$ is assigned to each arc $(i, j) \in A$ representing the travel cost between two customers. Let $C = \{1, \ldots, m\}$ be a set of homogeneous vehicles with capacity Q. Here we denote the vertex $i = 0$ representing the depot (launching

site). The CVRP consists of finding a minimum cost set of m routes starting and ending at the depot such that all customers are visited exactly once, all customers' demands are satisfied and the capacity of the vehicles are respected. The CVRP is well known to be $\mathcal{NP}$-hard. More information about the VRP and its variants can be found, e.g., in Golden and Assad (1988), Cordeau et al. (2007), Golden et al. (2008), Toth and Vigo (2002), Eksioglu et al. (2009), Lahyani et al. (2015) and Braekers et al. (2016).

The m-TSP is closely related to the VRP. In the m-TSP, m minimum cost tours starting at the depot must be found such that every vertex in $V \setminus \{0\}$ is visited exactly once. The m-TSP can be reduced to the CVRP if all vehicles are considered to have infinite capacity. An extensive literature review on models and algorithms for the m-TSP is presented by Bektas (2006).

The VRP and the m-TSP have been widely studied for terrestrial applications, but with the development of new technologies, such as unmanned vehicles, new variants of these problems are gaining interest among the scientific community. The problem of routing an aerial vehicle is more complex than the VRP, because it combines the combinatorial characteristics of the VRP with the complexity of dealing with the dynamical systems of UAVs (i.e., flight dynamics, battery life, wind conditions).

2.3.1 UAV task assignment problem

The UAV Task Assignment Problem (UAVTAP) consists of finding an optimal assignment of UAVs to a set of tasks. Often, the UAVs have different characteristics and the tasks present particular requirements. It has been shown that this problem is $\mathcal{NP}$-hard (Alidaee et al., 2010). Due to the quick development of UAV technology, new challenging assignment problems arise every day and many algorithms have been developed to address the new challenges. Figure 2.2 shows the number of publications by year in UAVTAPs, the y axis corresponds to the number of publications found in the Science Direct and Web of Knowledge databases. One can observe that this field of research has gained attention by the scientific community. A detailed literature review about algorithms for multi-robot TA problems can be found in Khamis et al. (2015).

The UAVTAP shares many similarities with the VRP. Many examples in the literature support this claim. One can cite, for instance, TA with time windows (Karaman and Inalhan, 2008), multi-depot (launching points) (Darrah et al., 2012), task allocation with resource constraints (Kim et al., 2015), TA with flexible demand (Alidaee et al., 2011), real-time and dynamic assignment (Kim et al., 2007; Lin et al., 2013), time-dependent TA (Kingston and Schumacher, 2005), and finally, TA under uncertainty (Alighanbari and How, 2008; Hu et al., 2015a). Nonetheless, new features have been also introduced, e.g., the possibility of heterogeneous UAVs to perform multiple operations at the same time (Shima and Schumacher, 2009). On the other hand, the kinematics and dynamics

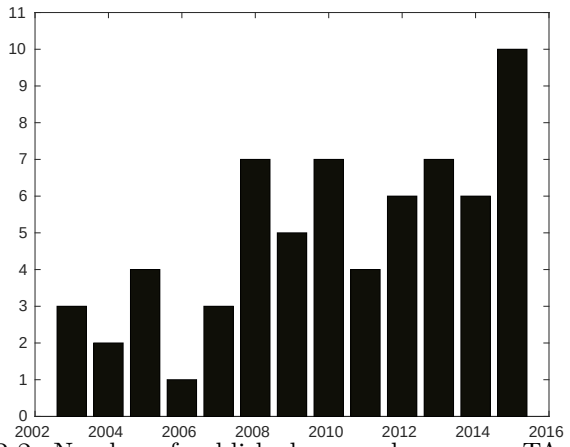


FIGURE 2.2: Number of published papers by year on TA problems.

of the UAVs are usually not considered, as opposed to the formulation of path/motion planning and TO problems.

2.4 UAVRTOP taxonomy

In this section, a taxonomy is proposed in order to help the readers to identify the key differences among various UAV routing problems and guide their research towards the development of new algorithms.

We have identified 20 attributes that are common in the UAV TO, PP, routing and TA literatures. They define common features of UAVs routing problems, such as the kind of fleet, the mission characteristics, the flying dynamic assumptions, etc. Attributes are further grouped into five classes. The first class collects the characteristics of the UAVs, the second class represents the characteristics of the waypoints, the third class describes the characteristics of the environment, the fourth class involves the characteristics of the launching point(s) and the last class is concerned about flight duration. These are listed in Table 2.1. The last two lines of this table are not part of the taxonomy, but they are important for the understanding of the A.1.

In the class **UAVs**, the fleet and how the UAVs' kinematics and dynamics are modelled are considered as follows:

1. *Multiple* - A fleet of vehicles is available (as opposed to a single vehicle).
2. *Heterogeneous* - Heterogeneous fleet, i.e., the fleet is composed by vehicles with different characteristics (as opposed to a homogeneous fleet). For example, vehicles with different flight endurances, different payload capacities and different flight dynamics.
3. *Fleet Size* - The size of the fleet must be optimised (as opposed to a fixed fleet size).

4. *Capacity* - Vehicles are capacitated (as opposed to uncapacitated vehicles). A capacitated UAV might have, for example, a maximum flight endurance or maximum payload capacity.

5. *Geometric* - The vehicle's flight dynamics are neglected (as opposed to considering flight dynamics). For example, by using Euclidean distances between waypoints instead of flight distances.

6. *Dubins'* - Dubins' vehicles are used to model the UAVs. A Dubins' vehicle can be defined as a vehicle with limited turning radius, i.e., it is unable of performing sharp turns. This approach has been extensively used for modelling ground and air vehicles moving in two dimensions. However, this model is quite simplistic and often not suitable for more complex UAVs.

7. *EOM* - A set of differential equations has been used to model the vehicles' kinematics and dynamics (as opposed to neglecting the dynamics). This approach is more often employed for modelling UAVs in complex environments, such as open fields under the influence of wind and operations were accurate trajectories are necessary.

The class **Waypoints** presents the attributes of the waypoints (vertices):

8. *Multiple* - Multiple waypoints must be visited (as opposed to a single waypoint or destination).

9. *Unordered* - The visiting order of waypoints is unknown (as opposed to a predefined order). This makes routing decisions necessary.

10. *Visits* - The waypoints can be visited multiple times (as opposed to a single visit for each waypoint).

11. *Constraints* - Special mission constraints must be considered. For instance, time-windows, precedences, and special boundary conditions.

12. *Covering Region* - A continuous, but not necessarily convex, region (or airspace) is defined over the waypoint. We believe this characteristic is important to the UAVRTOP since most UAVs' sensors require at least a minimum radius of action in order to be effective. For example, camera-equipped UAVs become effective within a certain range from the object that needs to be photographed. The same applies for UAVs carrying wireless network sensors.

The **Environment** class collects the attributes about the environment where the UAVs operate:

13. *3D* - The UAVs operate in a 3D space (as opposed to a 2D space).

14. *Obstacles* - The problem includes the presence of fixed or moving obstacles (as opposed to a obstacle-free environment). For example, a fleet of UAVs moving within a city must avoid collisions with the surrounding buildings and other UAVs.
15. *Wind* - The effects of wind are considered (as opposed to neglecting the wind effects).
16. *Real-time* - The problem must be solved in real-time. For example, waypoints and tasks arriving at random times and locations.

The class **Launching** groups the attributes about the number of launching points (depots):

17. *Multiple* - There are multiple launching points (as opposed to a single launching point).
18. *Inter-depot* - There are inter-depots available (e.g., for refuelling, battery replenishment or maintenance of the UAVs).

Papers are classified in class **Time** according to the way the flight time is considered in their models:

19. *Fixed* - The UAVs' flight times between arcs can be computed beforehand. This is a common characteristic of some PP methods (e.g., the Dubins' model).
20. *Variable* - The UAVs' velocities and flight times between arcs are optimisation variables.

In order to provide a survey of the most relevant and recent papers, we adopted the following procedure. Papers published since 2010 were collected from the following databases: *The Web of Science, Google Scholar* and *ScienceDirect*. We have limited our search to papers published in English. In order to cover the most common types of UAVs, we considered Unmanned Combat Aerial Vehicle (UCAV), Unmanned Aerial Systems (UAS) and *aerial gliders* in our search. The following keywords were used:

- UAV/UCAV/UAS/aerial glider trajectory optimization
- UAV/UCAV/UAS/aerial glider PP
- UAV/UCAV/UAS/aerial glider motion planning
- UAV/UCAV/UAS/aerial glider task assignment
- UAV/UCAV/UAS/aerial glider routing

Papers that focus on *Control Theory* for UAVs were not reviewed.

TABLE 2.1: Characteristics of the problems considered in this literature review.

	UAVs	
1	Multiple	A fleet of vehicles is available
2	Heterogeneous	The fleet is heterogeneous
3	Fleet size	The size of the fleet must be optimised
4	Capacity	Vehicles are capacitated
5	Geometric	The vehicle's flight dynamics are neglected
6	Dubins'	A Dubins' vehicle model has been used
7	EOM	A set of EOM is used to model the UAVs' flight dynamics
	Waypoints	
8	Multiple	Multiple waypoints must be visited
9	Unordered	The visiting order of waypoints is unknown
10	Visits	Waypoints can be visited multiple times
11	Constraints	Special mission constraints must be considered. (e.g., time-windows and boundary conditions)
12	Covering Region	If there is a continuous covering region around the waypoints
	Environment	
13	3D	The UAVs operate in a 3D space
14	Obstacles	If obstacles are present
15	Wind	The effects of wind are considered
16	Real-time	The problem must be solved in real-time
	Launching (Depot)	
17	Multiple	There are multiple launching points
18	Inter-depot	There are inter-depots available
	Time	
19	Fixed	The UAVs' flight times between arcs are known.
20	Variable	Flight times and velocities are optimisation variables
	Approach	The type of algorithm used to solve the problem (A.1)
	Application	A real-world motivation to solve the problem (A.1)

2.5 Critical review of the recent literature

In this section, we apply our taxonomy to 70 articles published between 2010 and 2016. We have balanced our analysis by considering articles dedicated to UAV TO/PP and UAV routing/TA. Papers devoted to technical and theoretical aspects of UAV flight dynamics were excluded from our analysis. Articles published in journals and conferences have been included in a number that we consider to be representative. Nonetheless, we apologise for any inadvertent omission of relevant papers.

The selected papers have been organised into Table 2.2. Each line of this table corresponds to one article and the meaning of each column relates to the numbering in Table 2.1. Each time an attribute is present in a paper the respective column is marked with "✕". Therefore, an empty cell indicates that its corresponding paper has not addressed the attribute indicated by this cell's column. A table with a detailed description of methods and applications for each article can be found in the A.1. Statistics about the Table 2.2 are provided in Table 2.3.

Three types of articles can be identified in Table 2.2. Papers focusing on UAV routing and TA can be identified by the presence of attributes 8 and 9. The second type, which involves papers on UAV PP and TO, exclusively, correspond to the ones where attribute 9 is absent. The third type consists of articles that integrate UAV routing and PP or

TO, which can be identified by the presence of attributes 5, 6 or 7 together with 8 and 9.

In Table 2.2, one can notice that 70% of the articles considered a fixed flight time. This indicates that most of the UAV literature is concerned with routing and PP algorithms, where constant velocity along the trajectories is a common assumption. The EOM of the vehicles where employed in 17.1% of the articles. In 53.8% of the papers on PP that applied a Dubins' model (which consist of only 18.6% of the total number of papers), the flight time has been considered as a variable.

Multiple UAVs were considered in 25% of the papers dealing with TO and PP. An interesting fact arises counting the number of papers dealing with multiple UAVs and their EOM. There seems to be a preference for using PP methods and the Dubins' model when a fleet of UAVs is taken into account. One can notice that the preferred strategy is to simplify the physical models of the UAVs so as to make the problem of designing multiple flyable routes more tractable. This happens in 44.4% of the articles on UAV PP and TO and in all the articles on UAV routing and TA.

Around 37.5% of the papers on TO and PP problems dealt with visiting multiple waypoints. However, only 14.3% attempted to integrate PP and TO to routing decisions. Among them, 3 papers employed the UAVs' EOM. This gives an indication that integrated routing and TO is yet to be fully investigated in the literature.

Regarding environmental conditions, 40% of the papers have studied three dimensional problems. Obstacle avoidance was tackled in 22.8% of the articles. Only a few studies (10%) included the effects of the wind in the UAVs' trajectories. In addition, only 7.1% of the papers studied real-time applications.

In 78.5% of the papers articles focusing on UAV routing and TA, a fleet of UAVs was considered. A large amount (85.7%) of the articles on routing and TA have either neglected or simplified the dynamics of the UAVs. Approximately 18% of the articles have modelled the UAVs as Dubins' vehicles. There is some overlap between these papers since some of them employ more than one methodologies. This suggests the preference for simplified vehicle models when dealing with UAV routing.

Table 2.3 illustrates other differences between the literature on UAV TO/PP and UAV routing/TA. Each row of Table 2.3 shows four classes that were defined in the proposed taxonomy and their respective frequencies (defined as the number of non-empty cells divided by the total number of cells in that class). For example, for the articles tackling TO and PP, the number of non-empty cells for class Depot is 3 and the total number of cells for the same class is 64. Hence the density of class Depot for TO/PP papers is $3/64 = 0.047$. One can notice that while the routing/TA literature is able to include more VRP-like attributes (like multiple UAVs and waypoints), the literature on TO/PP is more concerned about modelling environmental aspects. Including environmental

TABLE 2.2: Summary of the taxonomic review on 70 selected papers.

Author(s)	**UAVs**							**Waypoints**						**Env.**			**Dep.**		**Time**	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Al-Sabban et al. (2012)							×								×					×
Babel (2011)				×										×					×	
Babel (2012)			××										×							×
Bae et al. (2015)			××					×											×	
Baiocchi (2014)				×				×					×						×	
Bandeira et al. (2015)				×				××				×							×	
Bednowitz et al. (2012)	×			×				××			×								×	
Besada-Portas et al. (2010)	×			×				×					×						×	
Besada-Portas et al. (2013)	×			×				×					×						×	
Casbeer and Holsapple (2011)	××			××				××			×						×		×	
Chakrabarty and Langelaan (2011)				×				×					×		×					×
Chen et al. (2016)	×						×	×			×			×						×
Choe et al. (2016)	×			×				×					×				×		×	
Cobano et al. (2013)	×			×				××			×		×				×		×	
Cons et al. (2014)					×			××											×	
Crispin (2016)	×						×						×						×	
Dilão and Fonseca (2013)							×						×							×
Edison and Shima (2011)	××				×			××											×	
Enright et al. (2015)	×				××			××			×	×					×		×	
Evers et al. (2014)				××				××			×					×			×	
Faied et al. (2010)	×				×			××			×						×		×	
Filippis et al. (2011)					×									×					×	
Forsmo (2012)	×						×	××									×		×	
Fügenschuh and Müllenstedt (2015)	××	××					×	××			×	×	×		×		×			×
Furini et al. (2016)					×			××			×			×					×	
Gottlieb and Shima (2015)	×				×			××			×			×			×			×
Guerriero et al. (2014)	×		××	×				××			×						×		×	
Han et al. (2014)			××											×					×	
Henchey et al. (2016)					×			××			×	×		×	×	×				×
Huang et al. (2016)	×			×				×						×					×	
Hu et al. (2015b)	×		××					××									×		×	
Jaishankar and Pralhad (2011)				×							×		×	×					×	
Jiang and Ng (2011)	×			×				××			×								×	
Kagabo (2010)					×			×					×						×	
Kivelevitch et al. (2016)	×			×				××									×		×	
Kumar and Padhi (2013)							×	×												×
Kwak et al. (2013)	××			×				××	×		×		×						×	
Levy et al. (2014)	××		××					××	×								×	×	×	
Liu et al. (2013)				×										×						×
Liu et al. (2016)				×				×					×						×	
Manyam et al. (2015)	×				×			××			×						×			×
Mersheeva (2015)	×			××				××	×		×						×	×	×	
Mufalli et al. (2012)	×			××				××											×	
Murray and Karwan (2010)	××			××				××			×						×		×	
Murray and Karwan (2013)	××			××				××	×								×		×	
Myers et al. (2016)					×			××						×		×	×			×
Nguyen et al. (2015)				×				××	×			×	×							×
Niccolini et al. (2010)	××			×				××			×		×						×	
Park et al. (2012)			××					××	×	×									×	
Pepy and Hérissé (2014)							×													×
Pharpatara et al. (2015)						×							×						×	
Rogowski and Maroński (2011)							×													×
Shanmugavel (2013)					×														×	
Silva et al. (2015)							×						×		×					×
Song et al. (2016)	××			××				×			×						×	×	×	
Stump and Michael (2011)	×			××				××	×	×	×						×		×	
Sundar and Rathinam (2014)				×		×		××			×		×					×	×	
Techy et al. (2010)					×										×					×
Thi et al. (2012)	×			××				××											×	
Vilar and Shin (2013)	×				×			××			×			×					×	
Wang et al. (2015)	××			××				××			×		×	×			×		×	
Wang et al. (2016)					×								×						×	
Wu et al. (2011)					×						×		×	×					×	
Xu et al. (2017)	×					×					×		×	×						×
Yakıcı (2016)	×			××				××											×	
Yang et al. (2015)	×		×		×			××	×				×				×		×	
Yomchinda et al. (2016)					×						×		×							×
Zhang et al. (2011)					×						×		×	×		×			×	
Zhang et al. (2012)							×	××				×	×		×	×				×
Zhang et al. (2014)	×				×			××			×	×					×			×

attributes (such as obstacles and wind) is usually possible when the UAVs physical models are integrated to the optimisation problem.

In addition, the number of articles using a fixed flight time between waypoints is higher

TABLE 2.3: Densities per class for each classification.

Class	TO/PP	routing/TA
UAVs	20.1%	37.8%
Waypoints	11.9%	61.4%
Environment	26.6%	9.8%
Depot	4.7%	30.4%

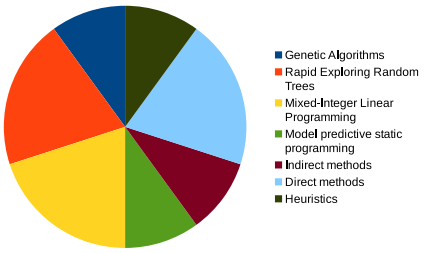
in the routing/TA literature (89.3%) than in the TO/PP literature (59.4%). This is also related to the preference of simplified physical models in the UAV routing/TA research community.

The analysed papers span a variety of different objective functions, according to the different applications that are considered. Minimising the total flight time or the overall travel distance is a common objective in the UAV routing literature. For delivery applications however, minimising delivery costs or the delivery time to customers is often preferred. In the case of powered UAVs, minimising energy expenditure or maximising the flight duration of each UAV is a common objective. On the other hand, for unpowered UAVs, one usually seeks to find a trajectory that maximises the flight range. For UCAVs, due to the high UAV unit costs and danger involved in military missions, minimising the risk of suffering an attack is usually acknowledged. For aerial survey operations, maximising area coverage is a popular objective. In task assignment problems, one often is looking for maximising service levels. Finally, for disaster assessment and response, minimising the total mission time or maximum flight duration (*makespan*) are one of the most adopted objectives.

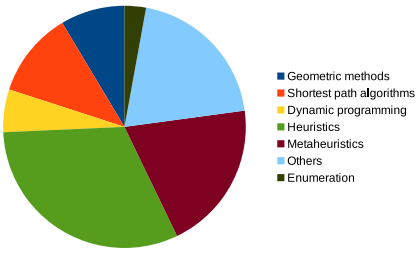
Figure 2.3 summarises the most employed algorithms for each research area described in Sections 2.2 and 2.3. These results are described in more details in the A.1. For the sake of simplicity, methods belonging to the same class of algorithms are grouped together on Figure 2.3. For example, the group *Metaheuristics* involves Iterated Local Search, Ant Colony Optimisation, Particle Swarm Optimisation, Evolutionary Algorithms, among others. Group *Heuristics* involves either methods combining more than one heuristic algorithm or specialised methods for a given problem. Group *Mixed-Integer Linear Programming* corresponds to exacts algorithms, i.e., branch-and-bound, column generation, among others. Group *Others* on Figure 2.3b represents different *ad hoc* path planning algorithms that did not fit into the previous categories, each algorithm being present in only one paper. These algorithms are often designed for one specific purpose and they are not based on classical methods. For example, category *Others* might include algorithms for coordinating the flow of UAVs in a dense airspace, algorithms based on splines for computing smooth paths, among others.

One can notice that, for articles involving TO (Figure 2.3a), exact methods for continuous optimisation are preferred. On the other hand, heuristics and metaheuristics are more frequent for the articles on UAV PP (Figure 2.3b). Heuristics, metaheuristics and

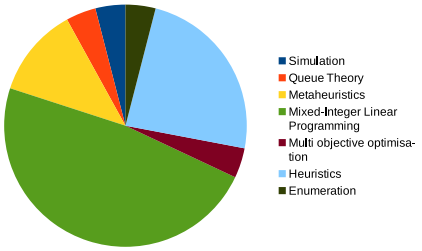
Mixed-Integer Linear Programming (MILP) algorithms are very popular among articles considering UAV routing and TA. Being exact methods more popular in the UAV routing papers (Figure 2.3c) and heuristics and metaheuristics more popular in the UAV TA articles (Figure 2.3d).



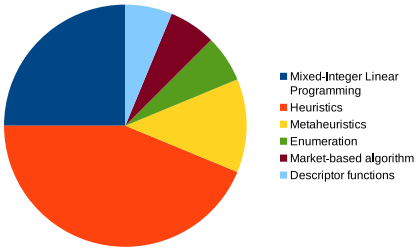
(A) Articles on UAV trajectory optimisation.



(B) Articles on UAV path planning.



(C) Articles on UAV routing.



(D) Articles on UAV task assignment.

FIGURE 2.3: Overview of methods and algorithms employed in the 70 selected papers.

2.5.1 Integrating routing and trajectory optimisation

Hereafter we highlight the contribution of articles that studied UAV routing and TO in an integrated framework. Such articles can be identified on Table 2.2 by attributes 7, 8 and 9 being marked with "✕". These papers present alternative frameworks to the UAVRTOP formulation presented in Section 2.1.

Zhang et al. (2012) investigated the problem of routing a combat 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 EOM 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 and 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 EOM 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 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.

2.6 Conclusions of the chapter

The UAVRTOP is a routing problem that takes into account the flight dynamics of UAVs. UAV routing problems usually ignore flight dynamics, while work on UAV trajectory optimisation usually ignores any routing aspects. Coupling these two important aspects leads to a more realistic approach that allows the design of optimal routes and trajectories for a fleet of UAVs flying simultaneously.

This problem arises from the current development of UAV technology and the vast number of applications that these vehicles can be used for. In this thesis, we first formalised the UAVRTOP. Next, an introduction to TOPs, VRPs and their variants has been provided. In addition, we introduced a taxonomy capable of classifying UAV routing/TA and UAV TO/PP problems according to their most relevant features. This taxonomy included 20 attributes that were common in the literature. Finally, we applied the proposed taxonomy to 70 recent papers.

The literature on UAVs routing problems has been surveyed and a lack of articles integrating UAV routing and TO has been identified. In particular, the UAVs' flight dynamics is often simplified or neglected. In many cases the behaviour of UAVs cannot be satisfactorily approximated only by their kinematics, like in the case of terrestrial robots (Forsmo, 2012). We believe that integrating the UAVs dynamical systems into routing problems is a key concept for complex operations. A realistic routing and TO algorithm must take into account the vehicle's kinematics and dynamics. In addition, by considering the UAVs' EOM one can also better approximate, for example, the vehicles' energy consumption, which is highly important for UAVs with limited battery duration. Modelling energy consumption is an issue that needs further investigation in the UAV routing literature.

Flight safety is an important aspect regarding the use of UAVs. Most research on UAV routing do not consider, for example, collision avoidance and wind conditions. This is important, e.g., for goods distribution within urban areas, where collisions with buildings and manned aircraft must be avoided and the fleet of UAVs must operate in a robust and reliable way.

Usually, research on UAV path and route planning concentrates on modelling kinematics. In many articles about UAV routing and TA even the kinematics are neglected. Models taking into account the forces acting on these vehicles, the interaction with wind ant their manoeuvring capabilities could though result in computationally expensive formulations, but such models might allow for more realistic solutions.

Mathematical formulations and algorithms capable of tackling complex unmanned aerial systems in a routing framework are recently appearing in the literature. A first step in this direction has been made by Zhang et al. (2012), Forsmo (2012) and Fügenschuh

and Müllenstedt (2015). Zhang et al. (2012) proposed a heuristic method, based on the 3DOF EOM of a UAV. Whereas Forsmo (2012) and Fügenschuh and Müllenstedt (2015) developed MILP formulations based on simplified dynamic equations. Only problems with limited size have been solved by the aforementioned authors. Therefore, the development of efficient frameworks for solving UAVRTOPs still raises challenging research questions that need to be answered.

Chapter 3

Glider Dynamics

In this chapter, the flight dynamics model that will be used throughout this thesis will be presented. In addition, we perform some computational experiments with the model and provide some insights on the trajectory optimisation of aerial gliders.

3.1 Introduction

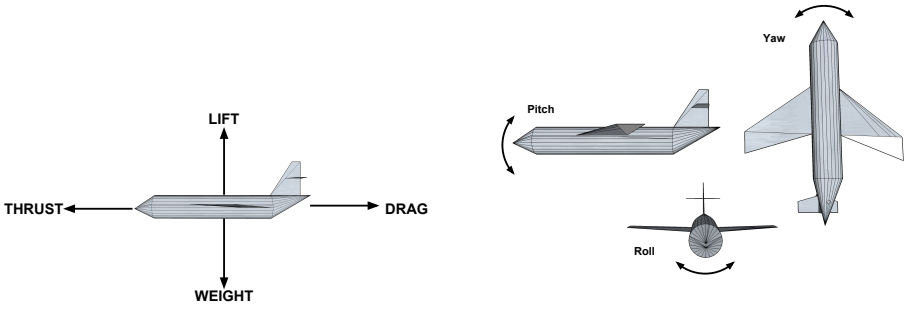
The four basic forces acting on an aircraft during flight are *thrust, drag, lift* and *weight*, as depicted in Figure 3.1a. Thrust is the force generated by the on-board propulsion of the aircraft itself. Drag is the air resistance acting upon the airplane's fuselage. Lift is the force generated by airflow through the control surfaces (flaps, ailerons, elevators and rudders), allowing the aircraft to fly. The weight models the force pulling the aircraft to the centre of the Earth. For a glider, thrust is absent, since there is no engine on board.

An aircraft is said to fly in *equilibrium* (a.k.a. steady-state flight) when the basic forces balance each other out. For instance, a powered steady-level flight can be achieved when lift equals weight and thrust equals drag. In simple terms, two types of equilibrium can be described, *static* and *dynamic*. Static equilibrium is related to the absence of velocity (static position). Dynamic equilibrium is related to the absence of acceleration (e.g., an object moving at constant velocity). In order to find steady-state conditions, we assume that gliders fly in dynamic equilibrium.

By activating the control surfaces of the aircraft, one can change its angular orientation. Angular orientation can be defined, for example, in terms of Euler angles, namely *pitch, yaw* and *roll*, with respect to a fixed reference frame. Alternative representations can also be applied, e.g., *quartenions* and *rotation matrices*, but they will not be considered in this work. Figure 3.1b depicts the planes of actuation of each Euler angle.

The control of an aircraft's horizontal orientation is usually described in terms of the Angle-of-Attack (AoA). The AoA is the difference between the pitch angle and the flight

path angle when the flight path angle is referenced to the atmosphere (Boeing, 2000)[1]. We assume that the AoA can be written as a function of the *lift coefficient* (C_L). This is a common assumption in the flight dynamics literature (Stengel, 2004). In simple terms, the lift coefficient describes the amount of lift generated by an aircraft's wings. We refer the interested reader to the books by Russell (1996) and Stengel (2004) for a more detailed understanding of aircraft flight dynamics.



(A) Aerodynamic forces. (B) Planes of actuation of the Euler angles.

FIGURE 3.1: Relevant forces and angles in an aircraft's flight.

3.2 Gliding flight techniques

Dynamic soaring is a nature-inspired powerless flight technique that takes advantage of wind gradients. This technique has been first studied by Rayleigh (1883) as an explanation for the flight of pelicans and other large birds. With the recent developments in UAV technology, this technique has become popular for autonomous gliding flight.

Several studies have acknowledged the use of autonomous gliders for different purposes. Zhao (2004), for example, studied optimal patterns for dynamic soaring. The most well known patterns can be defined as the *travelling pattern* (Figure 3.2a) and the *loitering pattern* (Figure 3.2b). A model for the flight of a glider is provided in the form of a set of EOMs. Next, several dynamic soaring flights are formulated as non-linear optimal control problems and then solved to optimality by means of a direct collocation method.

Langelaan (2007) and Chakrabarty and Langelaan (2011) employed a simpler aerodynamic model to optimise the trajectory of small UAVs. By assuming steady flight conditions, the authors we capable of computing long distance (and long duration) flights by combining a wind field prediction method with a trajectory planner.

Kagabo (2010) proposed a method for extending the flight of gliding vehicles by extracting energy from thermals, this technique is referred to as *static soaring*. By combining

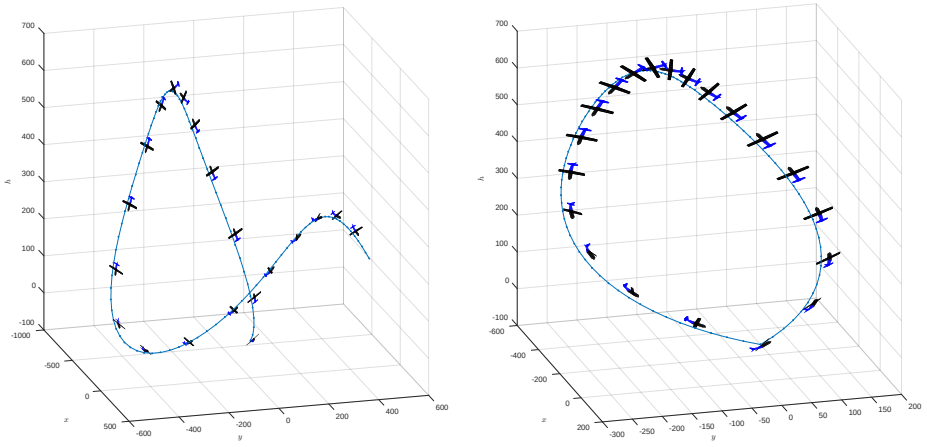
[1](Stengel, 2004, chap. 1,p. 3) defines the AoA as the aircraft-relative vertical angle between the centreline of the aircraft and the relative wind. In turn, the relative wind is defined as the aircraft's velocity relative to the surrounding air.

dynamic and static soaring the authors showed that the flight duration of multiple gliding UAVs could be extended. A similar approach is taken by Cobano et al. (2013) to optimise the trajectories of cooperative gliding UAVs.

Rogowski and Maroński (2011) considered the minimum-time problem for a glider moving in the vertical plane under given thermal conditions. In order to make the problem more tractable, the authors simplified the flight dynamics by assuming, for example, negligible aircraft dimensions and motion in the vertical plane only.

Wu et al. (2012) studies an application on emergency flight, in which the gliding trajectory of a commercial aircraft under engine failure must be optimised. The objective was finding a flyable descent trajectory for the aircraft in order to achieve a safe landing with certain speed and flight attitude.

Crispin (2016) proposes the use of balloon launched autonomous gliders for atmospheric research. The objective is to optimise the trajectory of a fleet of gliders for the uniform sampling of air within a given airspace and the efficient mapping of required physical and chemical properties.



(A) Travelling pattern of dynamic soaring. (B) Loiter pattern of dynamic soaring.

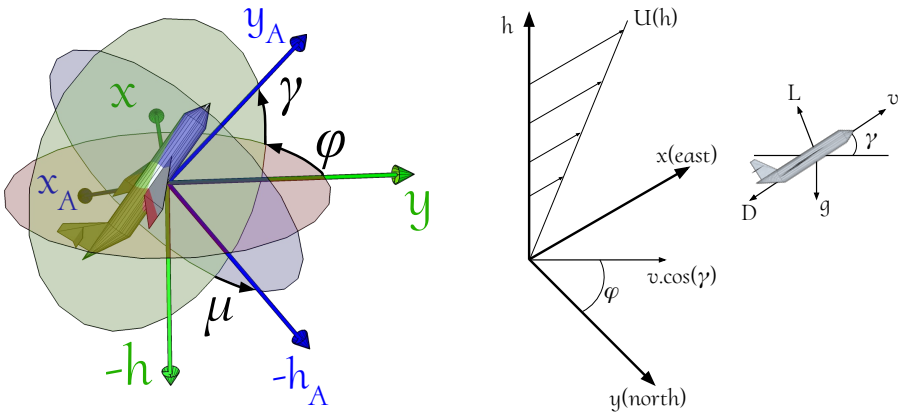
FIGURE 3.2: Patterns of dynamics soaring.

3.3 Gliders' equations of motion

The motion of an aerial glider can be modelled by a set of 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). The model presented by Zhao (2004) includes the following assumptions: the circumference of the earth is negligible compared to the range of flight, the density of the air can be considered constant, the wind is stationary and the mass of the glider does not change during the flight. Without loss of generality, one can assume that only the horizontal component of the

wind is present and that it can be described by a linear profile as a function of the flight altitude.

Let us define the *state* of a glider at time $\tau \in \mathbb{R}_{\geq 0}$ as a *state vector* $\mathbf{y}(\tau) = (x(\tau), y(\tau), h(\tau), v(\tau), \gamma(\tau), \varphi(\tau))^\top$, 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* $\mathbf{u}(\tau) = (C_L(\tau), \mu(\tau))^\top$, where $C_L(\tau) \in \mathbb{R}$ is the lift coefficient and $\mu(\tau) \in \mathbb{R}$ the bank angle. A North-East-Down coordinate frame is used to define the inertial position (Shaw-Cortez and Frew, 2015), while the airspeed is modelled in a wind relative frame (Deittert et al., 2009). In the following, the notation " $\dot{\cdot}$ " is used to represent time derivatives of the variables.



(A) NED frame and angles. (B) Flight model speeds and forces.
FIGURE 3.3: Coordinate frames and angles for the glider's dynamics.

The glider's physical model used in this thesis is based on the designs proposed by Bower (2010) and Flanzer (2012). These models are complimentary. From Bower (2010) we collected the parameters presented in Table 3.1 and the constants necessary for computing the *aerodynamic coefficient* (k_A). Based in Flanzer (2012) we defined reasonable lower and upper bounds for state and control variables. Both models have been tested and validated under different real flight conditions by their respective authors. We compute the k_A using Equation (3.1), see Kroo (2001). The *Oswald factor* e is computed using the Matlab function available in Sartorius (2013). Finally, $b_w = 2.49$ is the glider's wing span and S_w the wing area.

$$k_A = \frac{1}{\pi e \frac{b_w^2}{S_w}}. \tag{3.1}$$

The *wind strength* coefficient β is chosen so as to linearly approximate the average wind gradient profile for the UK, as suggested by Drew et al. (2013), for a reference altitude of

nearly 500 metres. The remaining model parameters and their meaning are summarised in Table 3.1.

The EOMs of a glider can be expressed by Equations (3.2–3.12). For the sake of notation simplicity, we omit the dependence on time (τ) from state, control and auxiliary variables.

TABLE 3.1: Environmental and glider constants

Symbol	Value	Description	Unity (IS)
ρ	1.22543	Density of the air at sea level	(kg/m^3)
g_e	9.80665	Gravity of Earth at sea level	(m/s^2)
β	0.02500	Wind strength	(s^{-1})
C_{D0}	0.01730	Coefficient of drag at zero-lift	(dimensionless)
k_A	0.03200	Aerodynamic coefficient	(dimensionless)
m_g	1.99000	Mass of the glider	(kg)
S_w	0.48500	Glider's total wing area	(m^2)

$$m_g\dot{v} = -D - m_g g_e \sin\gamma - m_g\dot{U}\cos\gamma\sin\varphi \tag{3.2}$$

$$m_g v\dot{\gamma} = L\cos\mu - m_g g_e\cos\gamma + m_g\dot{U}\sin\gamma\sin\varphi \tag{3.3}$$

$$m_g v\cos\gamma\dot{\varphi} = L\sin\mu - m_g\dot{U}\cos\varphi \tag{3.4}$$

$$\dot{x} = v\cos\gamma\sin\varphi + U(h) \tag{3.5}$$

$$\dot{y} = v\cos\gamma\cos\varphi \tag{3.6}$$

$$\dot{h} = v\sin\gamma, \tag{3.7}$$

where

$$D = \frac{1}{2}\rho S_w C_D v^2 \tag{3.8}$$

$$L = \frac{1}{2}\rho S_w C_L v^2 \tag{3.9}$$

$$C_D = C_{D0} + k_A C_L^2 \tag{3.10}$$

$$U(h) = \beta h \tag{3.11}$$

$$\dot{U} = \frac{dU(h)}{dt} = \beta v\sin\gamma. \tag{3.12}$$

In Equations (3.2–3.7), the wind's velocity is $U(h)$. The auxiliary variables D and L (Equations (3.8) and (3.9)) represent the drag and lift forces, respectively, acting on the glider.

By re-writing Equations (3.2–3.7) so that the time derivatives are isolated and by grouping the equations, one can obtain a compact representation of the system dynamics as follows:

$$\dot{\mathbf{y}} = f(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau), \tag{3.13}$$

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

3.4 Flight simulation

In this section, we validate the glider flight dynamics model presented in Section 3.3 by performing simulations of controlled flights. The simulations described here have been implemented and solved by means of Matlab's `ode45` function (version R2015b) in its standard configuration.

In our simulations, the inputs for `ode45` consist of the EOMs (3.13), initial state and control values $\mathbf{y}_o$ and $\mathbf{u}_o$, respectively, and a time span $[t_o, t_f]$. We apply a linear interpolation of the control variables within the given time span, where N is the discretisation size of the controls. In our experiments, the initial states are set as follows. The initial position (x_o, y_o, h_o) is set to $(0, 0, 1000)$ (in metres), the initial velocity is $v_o = 10\ m/s$, the initial flight path angle is set to $\gamma_o = 0$ degrees and the initial heading angle is $\varphi_o = 90$ degrees. The initial time t_o is set to 0 and the final time t_f equals 20 seconds. The first simulation consists of fixing the value of the first control variable to $C_L = 0.735$ and varying the value of the second control variable μ such as $\mu \in \{-2.5, 0, 2.5\}$. The discretisation size was set to $N = 300$.

Figure 3.4 shows the results of our first simulation. The values of the state variables are plotted on the vertical axis against time, which is plotted on the horizontal axis. The values of controls are plotted on the bottom of the figure. One can notice that the state variables present a very nonlinear response. Moreover, variables $y(t)$ and $\varphi(t)$ are the ones most affected by the change on μ. This can be explained by the fact that by changing the bank angle one controls the lateral movement of the glider, causing a change on its horizontal orientation.

Figure 3.5 shows the results of our second simulation. Now, the value of the second control variable is fixed to $\mu = 0$ and the first control variable C_L is chosen from $\{0.5, 0.735, 0.97\}$. In Figure 3.5, it can be noticed that the values of $y(t)$ and $\varphi(t)$ remain nearly constant, while the other state variables are affected by the different values of the lift coefficient. Again, most of the affected states present a very nonlinear response. Changes in the lift coefficient affect, for example, the altitude and velocity of the glider. The explanation for this phenomenon is quite intuitive. The lift coefficient represents how much lift is provided by the wings. If more lift is generated, the glider will tend to climb, therefore decreasing its airspeed. If less lift is generated, the glider will descend and gain airspeed with the help of gravity.

Figures (3.6a–3.6c) show the resulting flight trajectories for each different fixed values of μ. The glider's trajectory deviates to left and right according to the different values of the bank angle.

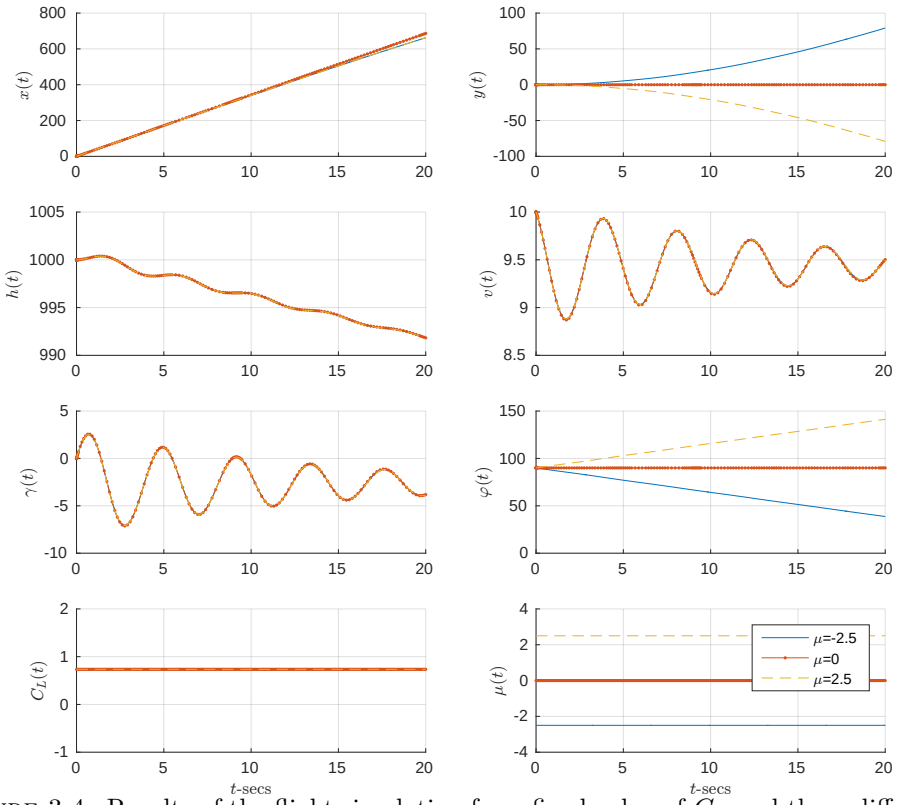


FIGURE 3.4: Results of the flight simulation for a fixed value of C_L and three different fixed values for μ.

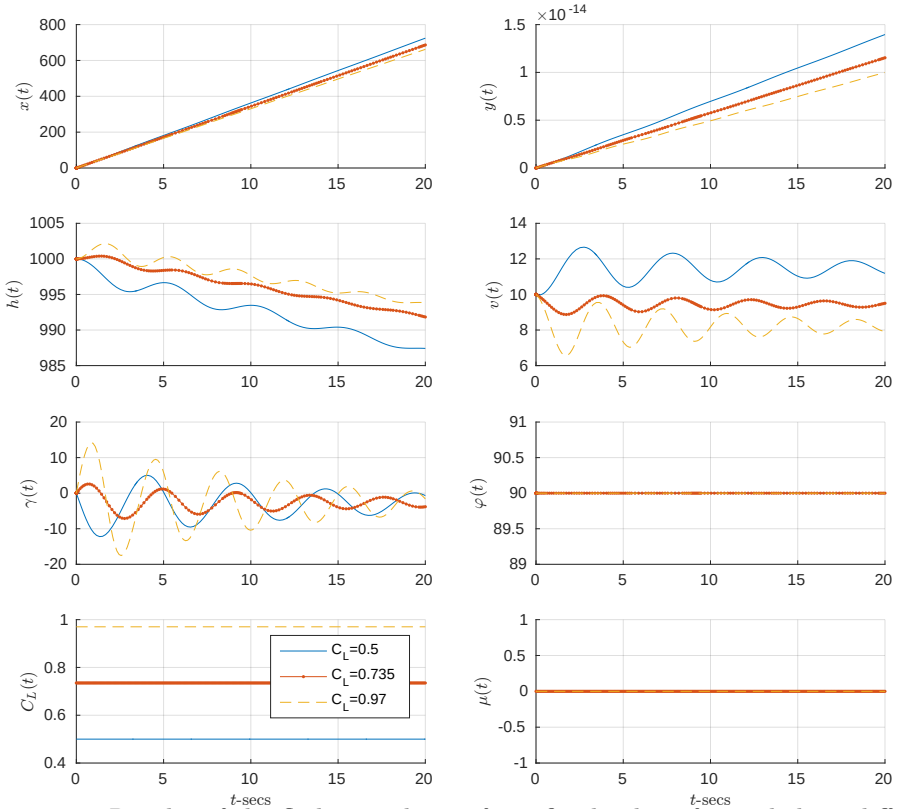
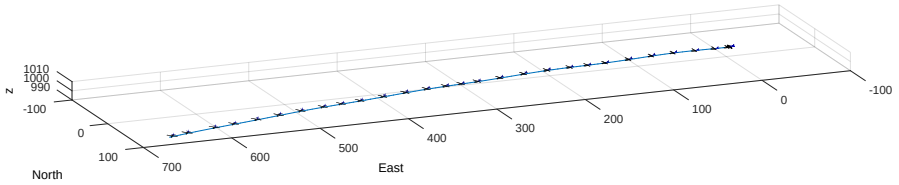
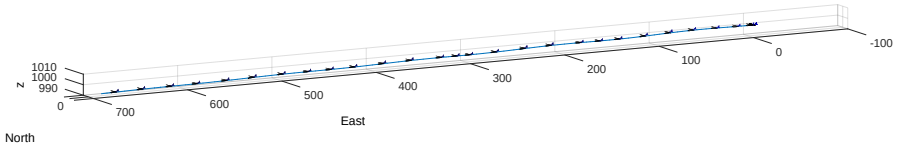


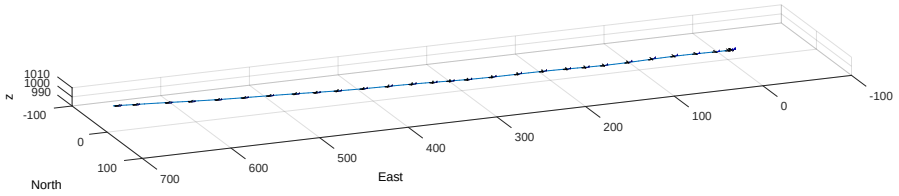
FIGURE 3.5: Results of the flight simulation for a fixed value of μ and three different fixed values for C_L.



(A) Trajectory for $\mu(t) = -2.5$.



(B) Trajectory for $\mu(t) = 0$.



(C) Trajectory for $\mu(t) = 2.5$

FIGURE 3.6: Resulting flight trajectories for different values of μ.

3.5 Glider trajectory optimisation

In this section we summarise the experiments and insights that were obtained by reproducing part of the work presented by Gao et al. (2014). The TO problem presented by Gao et al. (2014) consists of finding the minimum wind strength, denoted as β, required for an unmanned glider to fly in two different patterns, namely, the loiter pattern (Figure 3.2a) and the travelling pattern (Figure 3.2b).

The glider trajectory optimisation problem studied by Gao et al. (2014) can be more precisely stated as follows. Let us denote by $\mathbf{y}(\tau)$ and $\mathbf{u}(\tau)$ the state and control variables, respectively, of a glider moving according to the EOMs (3.13). States and controls are assumed to be lower bounded by $\mathbf{y}_{lb}$ and $\mathbf{u}_{lb}$, and upper bounded by $\mathbf{y}_{ub}$ and $\mathbf{u}_{ub}$, respectively. The independent variable τ, which represents time in our case, is assumed to lie in the closed interval $[\tau_o, \tau_f]$. Without loss of generality, we assume that the initial flight time τ_o equals 0. The final flight time τ_f is not necessarily know in advance and may be treated as an optimisation variable. Desired initial and final states are defined as $\mathbf{y}_o$ and $\mathbf{y}_f$. The glider trajectory optimisation presented by Gao et al. (2014) can be defined as in Equations (3.14–3.21).

$$\min_{\mathbf{u}(\tau)} \quad \beta \tag{3.14}$$

$$\text{s.t.} \quad \dot{\mathbf{y}}(\tau) = f(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau) \tag{3.15}$$

$$\mathbf{y}(\tau_o) = \mathbf{y}_0 \tag{3.16}$$

$$\mathbf{y}(\tau_f) = \mathbf{y}_f \tag{3.17}$$

$$\mathbf{u}_{lb} \leq \mathbf{u}(\tau) \leq \mathbf{u}_{ub} \tag{3.18}$$

$$\mathbf{y}_{lb} \leq \mathbf{y}(\tau) \leq \mathbf{y}_{ub} \tag{3.19}$$

$$\beta \geq 0, \mathbf{y}(\tau) \in \mathbb{R}^6, \mathbf{u}(\tau) \in \mathbb{R}^2 \tag{3.20}$$

$$\tau \in [\tau_o, \tau_f], \tau_o, \tau_f \in \mathbb{R} \tag{3.21}$$

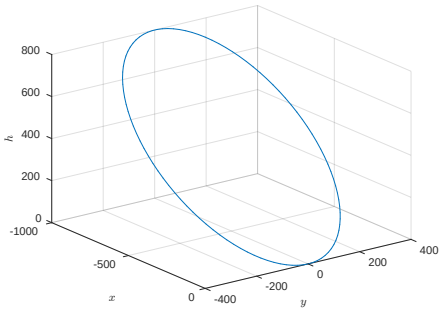
In the optimisation problem defined by Equations (3.14–3.21), constraint (3.15) defines the dynamic system (3.13). Equations (3.16) and (3.17) denote the initial and final conditions of the state variables. For the loitering pattern it is assumed that $\mathbf{y}(\tau_o) = \mathbf{y}(\tau_f)$. Constraints (3.18) and (3.19) represent lower and upper bounds over state and control variables. Finally, Expressions (3.20) and (3.21) represent the domain of the optimisation variables.

For solving the aforementioned TO problem, a direct collocation method was employed. Direct collocation methods work by discretising the EOMs into a discrete time index set within a given time interval. Next, the discretised EOMs can be embedded into a Nonlinear Programming (NLP) formulation that may be solved by any available nonlinear optimisation software. Methods for solving TO problems will be discussed in more details in Section 4.2.

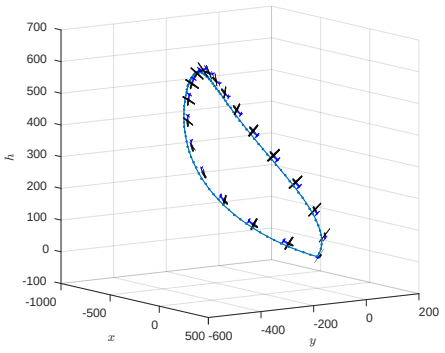
The nonlinear solvers IPOPT and WORHP were used to solve the resulting NLP problem, through the interface provided by the AMPL modelling language. Both software were capable of finding a local optimal solution to the problem within 15 seconds provided that a good initial guess was given before the optimisation. Otherwise, both solvers were incapable of finding a feasible solution. Figure 3.7 shows the initial guesses for the state variables, control variables and the glider's trajectory in the left hand side. The optimal solution is shown in the right hand side of Figure 3.7. One can notice that, in order to achieve convergence to a local optimal solution, the solvers required initial guesses that were already close to the optimal one.

3.6 Conclusions of the chapter

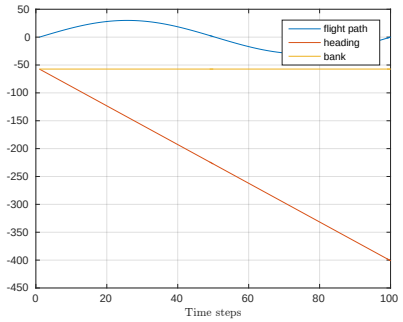
In this chapter, we introduced basic concepts about flight dynamics such as basic aerodynamic forces and steady-state flights. In addition, we provided an introduction to gliding flight techniques. However, the main purpose of this chapter consisted of to presenting the gliding flight dynamics model that well be used throughout this work. With this purpose in mind, we introduced the coordinate frames used to write the glider's EOMs. Next, the set of ODEs proposed by Zhao (2004) was presented, together with the physical parameters of the glider model that was adopted in this thesis. We performed two computational simulations with the the glider's EOMs, in order to gain some insights and to validate the implementation of the model.



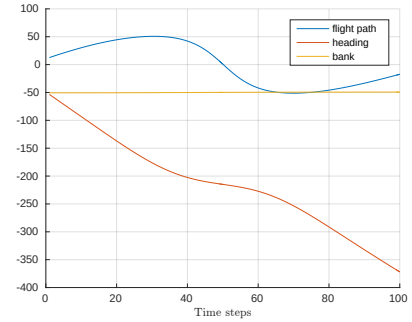
(A) Initial guess for the trajectory.



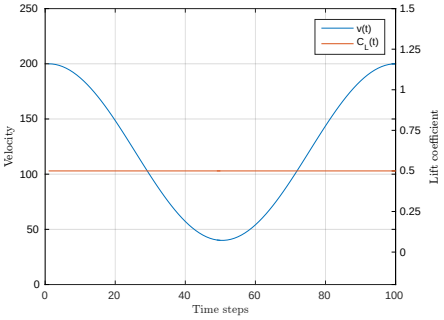
(B) Optimal trajectory.



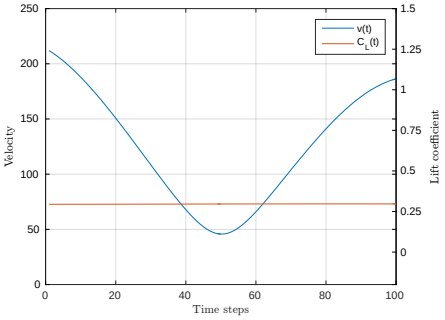
(C) Initial guess for the angles.



(D) Optimal angles.



(E) Initial guess for $v(t)$ and $C_L(t)$.



(F) Optimal $v(t)$ and $C_L(t)$.

FIGURE 3.7: Comparison between the initial guesses (left) and the local optimal solution (right).

With the purpose of obtaining additional insights on the optimisation of a gliding flight we reproduced part of the results of the paper by Gao et al. (2014). More specifically, we studied the problem of optimising the trajectory of a glider performing a dynamic soaring flight. A direct collocation method was applied for solving this TO problem. However, embedding the glider's EOMs into an optimisation problem leads to a nonlinear and nonconvex formulation. Two nonlinear optimisation solvers were used to solve the resulting NLP problem. Nonetheless, both solvers failed to return a feasible solution unless a good initial guess was provided beforehand. This suggests the use of an alternative approach to tackle the glider's dynamics.

Chapter 4

The Glider Routing and Trajectory Optimisation Problem

In this chapter, we introduce the Glider Routing and Trajectory Optimisation Problem (GRTOP). Next, a background on direct methods for trajectory optimisation is provided. We propose a MINLP formulation for the GRTOP and several solution strategies are discussed. Finally, the results of a number of computational experiments are presented.

4.1 Introduction

The Glider Routing and Trajectory Optimisation Problem (GRTOP) consists of finding optimal routes and trajectories for a fleet of unmanned aerial gliders with the task of visiting a set of locations. Gliders are UAVs without on-board propulsion. This problem arises from disaster assessment applications, in which camera-equipped gliders survey a number of risky locations in a post-disaster situation. The collected information can be used to assess the severity of the effects in the aftermath of a disaster. This problem was motivated by discussions with the Royal National Lifeboat Institution, the leading UK charity in providing flood rescue response, among other services.

Controlled powered drones have been used for raising emergency response in several scenarios, see for example, Chowdhury et al. (2017) and the recent Grenfell Tower disaster (Laville et al., e 15). However, these drones are expensive and often require experienced pilots to be operated. In aerial survey operations, a fleet of camera-equipped low cost balloon-launched autonomous gliders (Crispin, 2016) can be 3D printed (Keane et al., 2017) and do not require a specialised team to be operated. We believe this solution concept allows for rapid response to disasters.

Many Trajectory Optimisation Problems (TOPs) are non-convex in nature, such as most of the aerospace engineering-related problems (Conway, 2010; Shaw-Cortez and Frew,

2015). Several optimisation techniques based on OC and NLP have been developed to tackle TOPs. We refer the interested reader to Betts (2001) for a complete overview of those methods. NLP-based solution methods are known to be sensitive to initial guesses, i.e., convergence can be only ensured if a proper initialisation is provided (Zhao, 2004). Constructing a good set of initial guesses for flying vehicles often requires expertise on flight dynamics. In order to overcome such difficulties, the non-linear dynamics of UAVs is often linearised, see, e.g., Hajiyev et al. (2015), How et al. (2015) and Harris and Acikmese (2013).

In this chapter, we propose a single-phase MINLP formulation for the GRTOP. Mixed-Integer Programming (MIP) has been already applied for solving TOPs and OC problems, see, e.g., Keviczky et al. (2008), Soler et al. (2014), Fügenschuh and Müllenstedt (2015), Yuan et al. (2015) and Maolaaisha (2015). In Chapter 2, we provide a detailed description of other papers dealing with routing and trajectory optimisation in an integrated framework. Our formulation simultaneously optimises routes and the flight trajectories along these routes. The flight dynamics of the fleet of gliders are modelled as dynamical constraints, while the field of view of the cameras is modelled as second-order cone constraints. We avoid dealing with non-convex dynamical constraints by linearising the gliders' EOMs, reducing the proposed MINLP into a MISOCP problem. To allow for a more tractable formulation while keeping high quality solutions, we relax the resulting linear dynamical constraints and add a corresponding penalisation term to the objective function. Next, we study several integration methods for solving the EOMs. Motivated by our application, we consider a number of real-life instances based on flood risk maps of cities in the UK as well as 180 randomly generated instances. We perform computational experiments to test alternative commercial MISOCP solvers and the performance of the integration methods when embedded into the MISOCP problem. Further computational experiments are performed to test different discretisation sizes and associated errors.

4.2 Direct methods for trajectory optimisation

In the literature, two main classes of numerical methods can be found for solving TOPs, namely, *direct* and *indirect* methods. In Section 2.2, an overview of these methods is presented. In this section, we aim at providing the reader with a basic understanding of the direct methods for solving TOPs.

Let us define a simple single phase TOP, which is constrained by boundary conditions at the final time τ_f. The vehicle's state and control variables at time τ can be represented by the vectors $\mathbf{y}(\tau)$ and $\mathbf{u}(\tau)$. Let $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}(\tau), \mathbf{u}(\tau))$ (Equation 4.2) represent the vehicle's dynamics. The objective function (4.1) minimises a measure of performance over the vehicle's states at time τ_f and the final time itself. The only operational constraint

(Equation 4.3) of this problem acts upon the boundary conditions at the final time. We assume that the initial conditions $\mathbf{y}(\tau_o)$ and $\mathbf{u}(\tau_o)$ at the initial time $\tau_o = 0$ are known *a priori*. This problem can be represented by the Equations (4.1-4.3).

$$\min \quad \phi(\mathbf{y}(\tau_f), \tau_f) \tag{4.1}$$

$$\text{s.t.} \quad \dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}(\tau), \mathbf{u}(\tau)) \tag{4.2}$$

$$\psi(\mathbf{y}(\tau_f), \mathbf{u}(\tau_f), \tau_f) = 0 \tag{4.3}$$

Direct methods rely on the discretisation of a infinite-dimensional TOP into a finite-dimensional one. This strategy is commonly known as "Discretise, then optimise".

In a *direct single shooting* method, for example, the controls are discretised on a fixed grid $\tau_o = \tau_0 < \tau_1 < \cdots < \tau_n = \tau_f$ using an arbitrary parametrisation scheme, where n is the number of discretisation points:

$$\mathbf{u}(\tau_j, \mathbf{a}) \approx \sum_{i=1}^{m} a_i \boldsymbol{\alpha}_i(\tau_j), j = 1, \ldots, n. \tag{4.4}$$

The functions $\boldsymbol{\alpha}_i, i = 1, \ldots, m$, need to be defined beforehand and the parameters $a_i, i = 1, \ldots, m$, become the optimisation variables. The corresponding states $\mathbf{y}(\tau_j, \mathbf{a}), j = 1, \ldots, n$, are regarded as dependent variables and can be found by numerical integration using the finitely many control parameters $a_i, i = 1, \ldots, m$. This way the feasibility of constraint (4.2) is guaranteed for the discretised controls.

The last step of the direct shooting method consists of solving the non-linear programming problem defined by Equations (4.5-4.6) in order to find the optimal vector of parameters $\mathbf{a} = (a_1, \ldots, a_m)$:

$$\min_{\mathbf{a}} \quad \phi(\mathbf{y}(\tau_n, \mathbf{a}), \tau_n) \tag{4.5}$$

$$\text{s.t.} \quad \psi(\mathbf{y}(\tau_n, \mathbf{a}), \mathbf{u}(\tau_n, \mathbf{a}), \tau_n) = 0. \tag{4.6}$$

In this section we described a simple direct method for solving TOPs. However, many other more sophisticated algorithms can be found in the literature. Reviewing such methods is considered out of the scope of this work. A new review of trajectory optimisation algorithms would not add any relevant contribution beyond what is presented by existing surveys. More information about algorithms for TO can be found, e.g., in the papers by Stryk and Bulirsch (1992), Betts (1998), Ross (2009), Wang (2009) and Rao (2014), and the books by Bryson (1975), Bertsekas (1979), Betts (2001), Bryson (2002) and Kirk (2012).

4.3 Problem definition

In the GRTOP, 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).

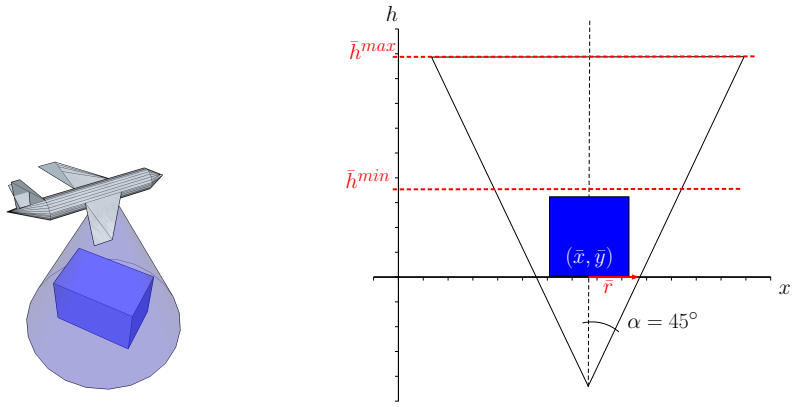
Each glider is equipped with a remote camera able to survey objects positioned within relative ranges. An inverted conic shape is adopted in order to model a cameras' field of view (Figure 4.1a). This type of geometry has also been used for UAV-camera systems in Ariyur and Fregene (2008), Roelofsen et al. (2016) and Nedjati et al. (2016a). 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 4.1b 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 $(\bar{x}_i, \bar{y}_i, \bar{r}_i, \underline{h}_i, \bar{h}_i), i \in V$, 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 $\underline{h}_i$ 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 $(x - \bar{x}_i)^2 + (y - \bar{y}_i)^2 \leq (h + \bar{r}_i)^2 \tan \alpha = (h + \bar{r}_i)^2$.

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.

4.3.1 A mixed-integer nonlinear programming formulation

In this section, a mathematical formulation for the GRTOP is proposed. In the following, we assume a fleet G of gliders to be available at a known launching point 0. Let V



(A) Camera model. (B) Waypoint geometry.

FIGURE 4.1: Modelling of the camera's opening angle and waypoint's geometry.

represent a set of waypoints that have to be visited and L a set of possible landing sites. We are asked to find optimal routes and trajectories for the gliders in G such that the total mission time is minimised.

The motion of each glider is constrained by the system of ODEs (3.13). We assume that the initial position, denoted by $\mathbf{x}_o$, of each glider is known in advance, i.e., $\mathbf{x}_g(\tau = 0) = \mathbf{x}_o, \forall g \in G$. The subvector $\mathbf{x}_g(\tau) = (x_g, y_g, h_g)$ represents the position of glider $g \in G$ at time $\tau \in [\tau_o, \tau_f]$, where $\tau_o = 0$ is the initial mission time and τ_f the maximum final mission time. We refer to the set of EOMs and initial conditions of each glider as their *dynamical system*.

We solve the GRTOP by means of *direct collocation* (Betts, 2001). In a direct collocation method, a continuous optimal control problem is discretised into a NLP by defining a grid of collocation points over a time interval $[\tau_o, \tau_f]$. Let us define N as the number of collocation points, where each time index t represents a time instant $\tau_t \in [\tau_o, \tau_f]$. From now on, a uniform time grid is adopted, as represented by $\tau_t = \tau_o + \eta t, t \in T, \eta = \frac{\tau_f - \tau_o}{N-1}$, where $T = \{0, \ldots, N-1\}$ is a set of collocation points (time index set). The value of η denotes the step size, which is constant in a uniform grid. Without loss of generality, we assume $\tau_o = 0$. In this work, we apply several different integration schemes to solve the dynamical system, namely, a forward Euler method, a Trapezoidal method, two Adams-Bashforth methods and two Runge-Kutta methods. They are discussed in more details in Section 4.4. More information about numerical methods for solving ODEs can be found, e.g., in the books by Betts (2001) and Hairer et al. (2011).

Let $\mathbf{y}_{gt}$ and $\mathbf{u}_{gt}$ approximate the continuous state and control vectors $\mathbf{y}_g(\tau_t)$ and $\mathbf{u}_g(\tau_t)$, respectively, of glider $g \in G$ at time index t. A simple method for approximating the continuous dynamical system of glider g at time index t can be written as in Equation (4.7), based on Euler's method,

$$\mathbf{y}_{g(t+1)} = \mathbf{y}_{gt} + \eta f(\mathbf{y}_{gt}, \mathbf{u}_{gt}, \tau_t), t \in T \setminus \{N-1\}. \tag{4.7}$$

Equations (4.7) describe one discretisation method that can be used to represent the non-linear gliders' dynamics in an optimisation problem, namely, the Euler's method. Changing the discretisation method, therefore, implies substituting the discrete system (4.7) by the corresponding equations of the new method. In our formulation, the dynamical system is relaxed by considering an error term $\varepsilon \in \mathbb{R}_{\geq 0}$, improving the feasibility of the set of dynamical constraints. Next, ε is added as a penalty to the objective function. This allows for a more tractable formulation, while maintaining the accuracy of trajectories. In Section 4.5, we show that the values of ε are small enough to be considered negligible in our examples.

We define the following binary decision variables:

$$a_{git} = \begin{cases} 1, & \text{if glider } g \text{ visits waypoint } i \text{ at time index } t \\ 0, & \text{otherwise.} \end{cases}$$

$$b_{git} = \begin{cases} 1, & \text{if glider } g \text{ lands in the landing site } i \text{ at time index } t \\ 0, & \text{otherwise.} \end{cases}$$

The GRTOP can be formally defined by the non-convex MINLP defined by Equations (4.8-4.31).

$$\min \quad \sum_{g\in G}\sum_{i\in L}\sum_{t\in T} t b_{git} + \varepsilon \tag{4.8}$$

$$\text{s.t.} \quad \sum_{\substack{g\in G \\ g\leq i}}\sum_{t\in T} a_{git} \geq 1, \forall i \in V \tag{4.9}$$

$$d_{git}^2 \geq (\bar{x}_i - x_{gt})^2 + (\bar{y}_i - y_{gt})^2, \forall g \in G, \forall i \in V, \forall t \in T \tag{4.10}$$

$$d_{git} \leq (h_{gt} + \bar{r}_i) + M(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T \tag{4.11}$$

$$h_{gt} \leq \bar{h}_i a_{git} + h_{ub}(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T \tag{4.12}$$

$$h_{g\tilde{t}} \geq h^{\min} a_{git} + h_{lb}(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T, \forall \tilde{t} \leq t \tag{4.13}$$

$$\gamma_{gt} \leq \hat{\gamma} a_{git} + \gamma_{ub}(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T \tag{4.14}$$

$$\gamma_{gt} \geq -\hat{\gamma} a_{git} + \gamma_{lb}(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T \tag{4.15}$$

$$\mu_{gt} \leq \hat{\mu} a_{git} + \mu_{ub}(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T \tag{4.16}$$

$$\mu_{gt} \geq -\hat{\mu} a_{git} + \mu_{lb}(1 - a_{git}), \forall g \in G, \forall i \in V, \forall t \in T \tag{4.17}$$

$$\sum_{i\in L}\sum_{t\in T} b_{git} = 1, \forall g \in G \tag{4.18}$$

$$\sum_{\substack{\tilde{t}\in T \\ \tilde{t}\leq t}} b_{gj\tilde{t}} \leq 1 - a_{git}, \forall g \in G, j \in L, i \in V, t \in T \tag{4.19}$$

$$r_{git}^2 \geq (\tilde{x}_i - x_{gt})^2 + (\tilde{y}_i - y_{gt})^2 + h_{gt}^2, \forall g \in G, i \in L, t \in T \tag{4.20}$$

$$r_{git} \leq \tilde{r}_i + M(1 - b_{git}), \forall g \in G, i \in L, t \in T \tag{4.21}$$

$$\mathbf{y}_{g,t+1} \leq \mathbf{y}_{gt} + \eta f(\mathbf{y}_{gt}, \mathbf{u}_{gt}, \tau_t) + \mathbf{1}\varepsilon, \forall g \in G, \forall t \in T \setminus \{N-1\} \tag{4.22}$$

$$\mathbf{y}_{g,t+1} \geq \mathbf{y}_{gt} + \eta f(\mathbf{y}_{gt}, \mathbf{u}_{gt}, \tau_t) - \mathbf{1}\varepsilon, \forall g \in G, \forall t \in T \setminus \{N-1\} \tag{4.23}$$

$$\mathbf{x}_{g0} = \mathbf{x}_o, \forall g \in G \tag{4.24}$$

$$\mathbf{y}_{lb} \leq \mathbf{y}_{gt} \leq \mathbf{y}_{ub}, \forall g \in G, \forall t \in T \tag{4.25}$$

$$\mathbf{u}_{lb} \leq \mathbf{u}_{gt} \leq \mathbf{u}_{ub}, \forall g \in G, \forall t \in T \tag{4.26}$$

$$a_{git} \in \{0,1\}, \forall g \in G, \forall i \in V, \forall t \in T \tag{4.27}$$

$$b_{git} \in \{0,1\}, \forall g \in G, \forall i \in L, \forall t \in T \tag{4.28}$$

$$d_{git}, r_{git} \in \mathbb{R}, \forall g \in G, \forall i \in V, \forall t \in T \tag{4.29}$$

$$\mathbf{y}_{gt} \in \mathbb{R}^6, \mathbf{u}_{gt} \in \mathbb{R}^2, \forall g \in G, \forall t \in T \tag{4.30}$$

$$\varepsilon \in \mathbb{R}_{\geq 0}. \tag{4.31}$$

The constants and "big-M" terms in the model are defined as follows. The M constant has been computed as the space diagonal of the smallest cuboid containing the waypoints, landing sites and launching point. This is an upper bound on the distance between a glider and any waypoint and landing site at any time. The value $h^{\min}$ is defined as the minimum allowed flight altitude before landing, i.e., $h^{\min} = \max_i\{\underline{h}_i | i \in V\}$, assuming that $h^{\min} < \min_i\{\bar{h}_i | i \in V\} - c$, with $c \in \mathbb{R}_{\geq 0}$ properly chosen. The values $\hat{\gamma} > 0$ and $\hat{\mu} > 0$ are small flight path and roll values forcing the glider to fly "flat" in the cone covering an object. Finally, we denote by $\mathbf{1}$ a vector of ones with the same length as $\mathbf{y}_{gt}$.

The objective function (4.8) accounts for a linear combination of the mission time and the error. The minimisation of the mission time forces the gliders to land as soon as possible. The second term of the objective function minimises the error that could be increased by landing too early. Constraints (4.9) state that every waypoint should be visited at least once. Preliminary tests showed that the formulation can be solved more efficiently by allowing an inequality rather than forcing equality in Constraints (4.9). In addition, revisiting waypoints does not necessarily cause an increase in the objective function value since Expression (4.8) accounts for landing times. Constraints (4.10) and (4.11) make sure that gliders fly within the cone above each waypoint in order to take pictures. Constraints (4.12) and (4.13) ensure that the gliders respect minimum and maximum surveying heights. Constraints (4.14) to (4.17) enforce gliders to be "flat" when taking pictures. Constraints (4.18) ensure that each glider lands in exactly one landing site and Constraints (4.19) make sure that gliders do not land before all waypoints are visited. Constraints (4.20) and (4.21) guarantee that gliders land within pre-assigned regions. We highlight that constraints (4.10) and (4.20) are second-order cone constraints. The dynamics of each glider are taken into account in Constraints (4.22) and (4.23) by applying a relaxation to Euler's method. Alternative formulations can be achieved by applying different discretisation methods. They will be presented in Section 4.4. By relaxing the dynamical system, we allow for a more tractable optimisation problem. Since the dynamical system is represented by a set

of non-convex ODEs, Constraints (4.22) and (4.23) are non-convex and the model is a MINLP. Constraints (4.24) define the initial positions of each glider and Constraints (4.25–4.26) define bounds on the state and control variables. Finally, Constraints (4.27–4.31) define the domain of the variables.

4.4 Linearisation and discretisation of the glider's dynamics

The model (4.8–4.31) combines routing and trajectory optimisation decisions in a non-convex formulation. Local optimisation software for non-linear optimisation often requires high quality initial guesses. In order to avoid this issue, we transform the MINLP into a more tractable convex model by linearising the gliders' EOMs. 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). In the following sections, we present the procedure for linearising the gliders' flight dynamics and the discretisation methods we applied for solving the resulting linear system.

4.4.1 Equilibrium flight and linearisation

A classic approach for linearising a system of ODEs consists of assuming the system operates in a *steady-state* condition, a.k.a. in equilibrium conditions. The equivalent linear system is then modelled assuming perturbations from this steady-state. Alternative techniques involve, for example, sequential and input-output linearisation. An interested reader can refer to the books by Russell (1996) and Stengel (2004) for more methods of finding steady-state conditions.

We denote by $\mathbf{y}_{eq}$ and $\mathbf{u}_{eq}$ the steady-states and their respective controls of the glider dynamics. In a steady flight, the resultant forces and moments acting on the vehicle are zero. In other words, let us define $\mathbf{y}_{eq} = (x_{eq}, y_{eq}, h_{eq}, v_{eq}, \gamma_{eq}, \varphi_{eq})$ and $\mathbf{u}_{eq} = (C_{L,eq}, \mu_{eq})$ as the state and control variables such that

$$\dot{\mathbf{y}} = f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau) = \mathbf{0}. \tag{4.32}$$

In order to find an analytic solution to Equation (4.32), the following assumptions are made as in Stengel (2004) and Langelaan (2007):

- Steady gliding flight, i.e., the equilibrium is achieved by matching the wind force with the drag, and the lift with the weight.
- The flight path and roll angles, γ and μ, respectively, are very small. Therefore, $\sin\gamma \approx \gamma$, $\cos\gamma \approx 1$, $\sin\mu \approx \mu$ and $\cos\mu \approx 1$. It is also assumed that $\varphi = 0$.

- The air mass is stable and the wind velocity is constant.
- The lift coefficient is constant.

From these assumptions, the EOMs (3.2–3.12) can be simplified to the following equations (4.33–4.35).

$$\dot{v} = -D/m_g - g_e\gamma = 0 \tag{4.33}$$

$$\dot{\varphi} = -L\mu/m_g v = 0 \tag{4.34}$$

$$\dot{\gamma} = L/m_g v - g_e/v = 0. \tag{4.35}$$

The optimal static lift coefficient is expected to minimise the drag-to-lift (D/L) ratio. Therefore:

$$\frac{\partial(D/L)}{\partial C_{L,eq}} = -\frac{C_{D0}}{C_{L,eq}^2} + k_A = 0 \implies C_{L,eq} = \sqrt{\frac{C_{D0}}{k_A}}. \tag{4.36}$$

From Equations (4.33–4.35), the expressions of the remaining equilibrium states are found:

$$v_{eq} = \sqrt{\frac{2m_g g_e}{\rho S_w C_{L,eq}}}, \tag{4.37}$$

$$\gamma_{eq} = -2\sqrt{k_A} C_{D0}. \tag{4.38}$$

Let us define the following new variables as perturbations around state and control variables as $\delta\mathbf{y}(\tau) = \mathbf{y}(\tau) - \mathbf{y}_{eq}$ and $\delta\mathbf{u}(\tau) = \mathbf{u}(\tau) - \mathbf{u}_{eq}$, respectively. Applying first order Taylor's expansion to the system (3.13) around the steady-state conditions and discarding the higher order terms gives:

$$T(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \delta\mathbf{y}, \delta\mathbf{u}, \tau) = f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau) + \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{y}}\delta\mathbf{y}(\tau) + \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{u}}\delta\mathbf{u}(\tau). \tag{4.39}$$

By definition, the first term of Equation (4.39) equals zero for the equilibrium condition. Matrices $A = \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{y}}$ and $B = \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{u}}$ denote the Jacobians of the dynamics (3.13) with respect to state and control variables. This linear system of ODEs can be re-written in state-space form as in Equation (4.40)

$$T(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \delta\mathbf{y}, \delta\mathbf{u}, \tau) = A\delta\mathbf{y}(\tau) + B\delta\mathbf{u}(\tau), \tag{4.40}$$

where the system's matrices A and B have been found by computing the derivatives of the glider's EOM (3.2–3.12) at the steady-state conditions $\mathbf{y}_{eq}$ and $\mathbf{u}_{eq}$:

$$A = \begin{bmatrix} 0 & 0 & 0.025 & 0 & 0 & 9.44023 \\ 0 & 0 & 0 & 0.99889 & 0.44456 & 0 \\ 0 & 0 & 0 & -0.04704 & 9.44023 & 0 \\ 0 & 0 & 0 & -0.09766 & -9.79579 & 0.01110 \\ 0 & 0 & 0 & 0.21947 & -0.04881 & 0.00006 \\ 0 & 0 & 0 & 0 & -0.02506 & 0 \end{bmatrix}, \mathbf{y}_{eq} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 9.45068 \\ -0.04705 \\ 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.62763 & 0 \\ 1.41127 & 0 \\ 0 & 1.03882 \end{bmatrix}, \mathbf{u}_{eq} = \begin{bmatrix} 0.73527 \\ 0 \end{bmatrix}.$$

In our approach, we approximate the nonlinear dynamical system defined in Equation (3.13) by the linear system of equations (4.40), such that:

$$\dot{\mathbf{y}} \approx A\delta\mathbf{y}(\tau) + B\delta\mathbf{u}(\tau). \tag{4.41}$$

4.4.2 Discretisation methods

In this work, we solve the gliders' EOMs by means of a *direct collocation* method (Betts, 2001). This is accomplished by discretising the linear EOMs represented by Equations (4.41) 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 EOMs is maintained. Due to the linearity of the EOMs, the MINLP formulation for the GRTOP, defined by Equations (4.8–4.31), is reduced to a MISOCP formulation that can be solved by commercial optimisation software without the need of initial guesses to converge.

4.4.2.1 Euler's method

The forward Euler's method is a first-order explicit numerical approach for solving ODEs (Hairer et al., 2011). Let $\mathbf{y}_t$ and $\mathbf{u}_t$ approximate $\mathbf{y}(\tau_t)$ and $\mathbf{u}(\tau_t)$, respectively, at time $\tau_t \in [\tau_o, \tau_f]$ such that:

$$\dot{\mathbf{y}} \approx \frac{\mathbf{y}_{t+1} - \mathbf{y}_t}{\eta}.$$

By using this approximation on Equation (4.41), we can write:

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \eta(A\delta\mathbf{y}_t + B\delta\mathbf{u}_t). \tag{4.42}$$

This dynamical system can be reformulated in terms of the discretised state and control variables in Equation (4.43). In a direct collocation method, these equations are used as constraints in an optimisation problem.

$$\mathbf{y}_{t+1} = (\eta A + I)\mathbf{y}_t + \eta B\mathbf{u}_t - \eta(A\mathbf{y}_{eq} + B\mathbf{u}_{eq}). \tag{4.43}$$

The feasibility of this dynamical system can be improved by considering an error ε and relaxing the equality in Equation (4.43) such that:

$$\mathbf{y}_{t+1} \leq (\eta A + I)\mathbf{y}_t + \eta B\mathbf{u}_t - \eta(A\mathbf{y}_{eq} + B\mathbf{u}_{eq}) + \mathbf{1}\varepsilon \tag{4.44}$$

$$\mathbf{y}_{t+1} \geq (\eta A + I)\mathbf{y}_t + \eta B\mathbf{u}_t - \eta(A\mathbf{y}_{eq} + B\mathbf{u}_{eq}) - \mathbf{1}\varepsilon. \tag{4.45}$$

We penalise ε in our objective function in order to maintain the quality of our solutions. Finally, we replace Constraints (4.22–4.23) by Constraints (4.44–4.45) when applying Euler's method for solving the dynamical system in our formulation. This strategy is repeated for all discretisation methods presented in Section 4.4.2.

4.4.2.2 Trapezoidal method

The Trapezoidal method is a second-order implicit approach for solving ODEs based on the trapezoidal rule for computing integrals (Hairer et al., 2011). The equation defining the trapezoidal method can be derived from both Runge-Kutta and Adams-Bashforth methods, and for the GRTOP it can be defined as follows:

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \frac{1}{2}\eta(A\delta\mathbf{y}_t + B\delta\mathbf{u}_t + A\delta\mathbf{y}_{t+1} + B\delta\mathbf{u}_{t+1}). \tag{4.46}$$

By re-writing this system in terms of the original variables, we find the following discrete linear system:

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \frac{1}{2}\eta(A(\mathbf{y}_{t+1} + \mathbf{y}_t) + B(\mathbf{u}_{t+1} + \mathbf{u}_t)) - \eta(A\mathbf{y}_{eq} + B\mathbf{u}_{eq}). \tag{4.47}$$

In order to use Equation (4.47) as a constraint in our formulation, we apply the same relaxation method as presented in Section 4.4.2.1.

4.4.2.3 Runge-Kutta methods

The Runge-Kutta methods are a family of numerical methods for solving initial value problems. They consist of sampling intermediate values between subsequent time steps in order to cancel out lower order error terms (Hairer et al., 2011). In this thesis, we apply a fourth-order Runge-Kutta method ($RK4$) in order to discretise the glider dynamics. The coefficients of the $RK4$ method for the linear glider dynamics are defined by Equations (4.48–4.51) in terms of the original discrete state and control variables.

$$k_1 = A(\mathbf{y}_t - \mathbf{y}_{eq}) + B(\mathbf{u}_t - \mathbf{u}_{eq}) \tag{4.48}$$

$$k_2 = A(\mathbf{y}_t + \frac{1}{2}\eta k_1 - \mathbf{y}_{eq}) + B(\hat{\mathbf{u}} - \mathbf{u}_{eq}) \tag{4.49}$$

$$k_3 = A(\mathbf{y}_t + \frac{1}{2}\eta k_2 - \mathbf{y}_{eq}) + B(\hat{\mathbf{u}} - \mathbf{u}_{eq}) \tag{4.50}$$

$$k_4 = A(\mathbf{y}_t + \eta k_3 - \mathbf{y}_{eq}) + B(\mathbf{u}_{t+1} - \mathbf{u}_{eq}), \tag{4.51}$$

where the auxiliary variable $\hat{\mathbf{u}}$ represents an interpolation of the control variables between time steps t and $t+1$. Here, two interpolation methods have been used. The first one consists of a linear interpolation (Equation (4.52)) and the second one consist of an exponential smoothing (Equation (4.53)), which weights the control history up to time step t.

$$\hat{\mathbf{u}} = \frac{\mathbf{u}_{t+1} + \mathbf{u}_t}{2} \tag{4.52}$$

$$\hat{\mathbf{u}} = \frac{1}{2} \sum_{k=0}^{k \leq t} \frac{1}{2^k} \mathbf{u}_{t-k} \tag{4.53}$$

Discretised EOMs can then be defined by the following equation:

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \frac{\eta}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{4.54}$$

In order to use Equation (4.54) as a constraint in our formulation, we apply the same relaxation method as presented in Section 4.4.2.1.

4.4.2.4 Adams-Bashforth methods

Linear multistep methods use information from previous steps to determine the current values of the state vector (Hairer et al., 2011). Unlike Runge-Kutta methods, multistep methods do not required interpolation of the control variables at intermediate steps since calculations are based on predetermined collocation points (in case of a direct collocation method). The Adams-Bashforth (AB) methods are a family of explicit integrators that compute the value of the current state from a linear combination of the values of previous states. In this article, a third-order Adams-Bashforth method ($AB3$) and a fourth-order Adams-Bashforth method ($AB4$) are presented, in terms of the original state and control vectors, in the form of Equations (4.55) and (4.56).

$$\begin{aligned} \mathbf{y}_{t+3} = \mathbf{y}_{t+2} + \frac{1}{12}\eta(&23(A(\mathbf{y}_{t+2} - \mathbf{y}_{eq}) + B(\mathbf{u}_{t+2} - \mathbf{u}_{eq})) \\ &-16(A(\mathbf{y}_{t+1} - \mathbf{y}_{eq}) + B(\mathbf{u}_{t+1} - \mathbf{u}_{eq})) \\ &+5(A(\mathbf{y}_t - \mathbf{y}_{eq}) + B(\mathbf{u}_t - \mathbf{u}_{eq})) \end{aligned} \tag{4.55}$$

$$\begin{aligned} \mathbf{y}_{t+4} = \mathbf{y}_{t+3} + \frac{1}{24}\eta(&55(A(\mathbf{y}_{t+3} - \mathbf{y}_{eq}) + B(\mathbf{u}_{t+3} - \mathbf{u}_{eq})) \\ &-59(A(\mathbf{y}_{t+2} - \mathbf{y}_{eq}) + B(\mathbf{u}_{t+2} - \mathbf{u}_{eq})) \\ &+37(A(\mathbf{y}_{t+1} - \mathbf{y}_{eq}) + B(\mathbf{u}_{t+1} - \mathbf{u}_{eq})) \\ &-9(A(\mathbf{y}_t - \mathbf{y}_{eq}) + B(\mathbf{u}_t - \mathbf{u}_{eq})) \end{aligned} \tag{4.56}$$

Unlike single step methods, the third- and fourth-order AB methods require 3 and 4 initial values at the first iteration, respectively. This can be accomplished by running a single step method in order to find these initial values and then continuing the solution process with a multistep AB method. In this chapter, we apply the Euler method in order to compute the initial values that are necessary for the Adams-Bashforth methods. In order to use Equations (4.55) and (4.56) as constraints in our formulation, we apply the same relaxation method as presented in Section 4.4.2.1.

4.4.2.5 Interpolation of the solution

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 (4.57).

$$\mathbf{u}(\tau) = \mathbf{u}_t + \frac{\tau - \tau_t}{\eta}(\mathbf{u}_{t+1} - \mathbf{u}_t) \tag{4.57}$$

The approximation for the system dynamics also follows a linear function (Equation (4.58)). Moreover, by integrating both sides of this approximation, as in Equation (4.59), one finds a quadratic interpolation for the state function, expressed by Equation (4.60). An appropriate choice for the integration constant is $c = \mathbf{y}_t$.

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\tau} = \mathbf{f}_t + \frac{\tau - \tau_t}{\eta}(\mathbf{f}_{t+1} - \mathbf{f}_t) \tag{4.58}$$

$$\int \mathrm{d}\mathbf{y} = \int \left(\mathbf{f}_t + \frac{\tau - \tau_t}{\eta}(\mathbf{f}_{t+1} - \mathbf{f}_t)\right) \mathrm{d}\tau \tag{4.59}$$

$$\mathbf{y}(\tau) = \mathbf{f}_t(\tau - \tau_t) + \frac{(\tau - \tau_t)^2}{2\eta}(\mathbf{f}_{t+1} - \mathbf{f}_t) + c \tag{4.60}$$

4.5 Computational experiments

The computational experiments described in the next sections have been implemented in the AMPL modelling language (version 20150516) and solved with CPLEX 12.7, Gurobi 7.0 and Xpress 8.0 on an Intel Core i7-4770 CPU with 3.40GHz and 16GB of RAM running under Linux Mint 17 64bits (kernel 3.13.0-24). These solvers were chosen due to availability of licenses and their popularity in the research community (Mittelmann, 2017). They were set to their standard configurations, with a time limit of 1 hour of execution each.

4.5.1 Generation of test instances

A number of test instances have been generated in the following way. First of all, we have considered instances having $n \in \{2, \ldots, 10\}$ waypoints and $m \in \{1, 2\}$ landing zones. The number of available gliders has been defined as $\lfloor n/2 \rfloor$, but limited to at most 3 gliders in our examples. Two classes of instances have been created. The so-called *small range* instances (represented by "S" in the instances' name) have been defined over an area of $1km^2$ and the so-called *medium range* instances (represented by "M" in the instances' name) over an area of $25km^2$. We assume a square shape for each area. In addition, each combination of number of waypoints and landing zones received 5 different random instances, so to have a diversity of test cases. These instances are grouped in our tables by the number of waypoints, e.g., the group GRTOP-S10 represents all small range instances with 10 waypoints.

The geometry of waypoints and landing zones has been defined by the parameters in Table 4.1. Let $\mathcal{U}[a, b]$ denote the continuous uniform distribution from a to b. The launching altitude h_o has been chosen from $\mathcal{U}[500, 600]$ for the small range instances and $\mathcal{U}[1000, 2000]$ for the medium range instances, with the values of the limits given in metres. In Table 4.1, the value of R represents the square's side of the area, $R = 1km$ for the small-ranged instances and $R = 5km$ for the medium range ones. All the other values in Table 4.1 are given in metres.

The positions of waypoints were not constrained. Overlaps were allowed except when they generate duplicates and were assigned randomly within the boundaries of the area. However, landing zones were not allowed to overlap with waypoints.

TABLE 4.1: Limits of parameters defining the geometry of waypoints and landing zones in the generated instances.

Parameter	a	b	Parameter	a	b
$\bar{x}$	0	R	$\tilde{x}$	0	R
$\bar{y}$	0	R	$\tilde{y}$	0	R
$\bar{r}$	10	25	$\tilde{r}$	10	25
$\underline{h}$	50	100	x_o	0	R
$\bar{h}$	200	300	y_o	0	R

4.5.2 Comparing the performance of different solvers

Table 4.2 shows a summary of the results for the small range instances. In this table, *Group* denotes the nine groups of 10 instances (organised according to the number of waypoints in each instance). The performance of each solver is shown on columns *CPLEX*, *Gurobi* and *Xpress*.

In Table 4.2, the *Status* column shows the tuple $(s_1, s_2, s_3) \in \mathbb{Z}^3$, representing the possible output statuses from AMPL as explained on AMPL (1998), where s_1 denotes the number of instances finished with status `solved`, s_2 represents the number of instances finished with status `solved?` and `limit` (i.e, with a gap greater than 0), and s_3 represents the number of instances finished with status `failure`. Column *UB* shows the average upper bound value per group. Note that the upper bound represents the objective function value of the incumbent solution, which is the best feasible solution found in the branch and bound search. Column *Error* corresponds to the normalised average error for each group. This has been calculated by averaging the error in each group and then dividing this average by the smallest error among the solvers for the same group. We use normalised average errors in order to make a comparison between different strategies easier. Column *Gap(%)* shows the average optimality gap at the end of the optimisation. Column *CPU(s)* represents the average computing time in seconds. Finally, column *Tree Size* shows the average number of explored branch-and-bound nodes for each group.

In order to test the solvers, we have chosen the Euler discretisation method. The number of collocation points N has been set to 30. The flight time horizon of each instance has been estimated by using the steady-state velocity and the largest range of that instance, i.e., $\tau_f = \max_{i \in V}\{\bar{x}_i, \bar{y}_i\}/v_{eq}$.

The solver Xpress outperforms the other solvers in most aspects. Best results overall in each column (except from the second) have been highlighted in boldface. Averages are shown for each solver. In general, Xpress produces the best objective function values. The relationship between CPU times and tree sizes indicates that Xpress is noticeably faster on processing the second-order cone relaxations during the branch-and-bound search. Due to a large number of failures and worse performance for computing Second-Order Cone Programming (SOCP) relaxations, CPLEX presents smaller average tree sizes on most cases. For our problem, Xpress has been capable of solving 76% of the instances to optimality as opposed to 32% and 24% of the instances that have been solved by CPLEX and Gurobi, respectively. One can notice that CPLEX has failed to find solutions to 15 problems in total. For the reasons exposed above, the solver Xpress has been chosen for the computational experiments presented in the next sections.

TABLE 4.2: Summary of the results for different solvers

Group	Status	UB	Error	Gap(%)	CPU(s)	Tree
			CPLEX			
GRTOP-S2	(10,0,0)	32.501	1.000	**0.00%**	22.348	1274.30
GRTOP-S3	(8,0,2)	**35.316**	1.007	**0.00%**	99.081	2428.75
GRTOP-S4	(2,0,8)	54.965	1.034	**0.00%**	1199.889	26176.00
GRTOP-S5	(3,4,3)	**54.687**	**1.000**	18.57%	2171.628	12652.43
GRTOP-S6	(3,6,1)	69.931	1.403	32.33%	2997.753	20993.56
GRTOP-S7	(1,8,1)	80.523	1.736	35.78%	3175.324	13763.67
GRTOP-S8	(2,8,0)	125.564	4.017	51.30%	3397.745	12404.38
GRTOP-S9	(0,10,0)	111.675	3.285	54.50%	3600.500	11439.10
GRTOP-S10	(0,10,0)	115.745	3.230	65.90%	3600.589	12096.90
avg.	-	75.66	1.97	28.71%	2251.65	12581.01
max	-	125.564	4.017	65.90%	3600.59	26176.00
			Gurobi			
GRTOP-S2	(10,0,0)	**32.500**	**1.000**	**0.00%**	44.125	**2831.40**
GRTOP-S3	(9,1,0)	36.909	1.028	2.70%	492.016	**7698.60**
GRTOP-S4	(1,9,0)	62.974	1.681	22.10%	3351.221	**40894.40**
GRTOP-S5	(1,9,0)	76.731	2.256	26.00%	3250.099	40628.50
GRTOP-S6	(1,9,0)	100.984	2.763	44.00%	3379.685	42566.10
GRTOP-S7	(0,10,0)	104.359	2.731	39.40%	3600.206	48218.50
GRTOP-S8	(0,10,0)	96.436	2.736	42.90%	3600.184	39544.80
GRTOP-S9	(0,10,0)	102.572	2.946	39.60%	3600.166	35009.00
GRTOP-S10	(0,10,0)	140.130	4.482	56.90%	3600.204	31395.40
avg.	-	83.73	2.4	30.4%	2768.66	32087.41
max	-	140.13	4.482	56.90%	3600.21	48218.50
			Xpress			
GRTOP-S2	(10,0,0)	**32.500**	**1.000**	**0.00%**	**7.124**	800.60
GRTOP-S3	(10,0,0)	35.810	**1.000**	**0.00%**	**14.024**	3250.30
GRTOP-S4	(10,0,0)	**50.073**	**1.000**	**0.00%**	**104.985**	18475.50
GRTOP-S5	(9,1,0)	55.652	1.020	**0.10%**	**727.232**	**186538.20**
GRTOP-S6	(7,3,0)	**60.125**	**1.000**	**2.40%**	**1294.716**	**140235.50**
GRTOP-S7	(9,1,0)	**62.825**	**1.000**	**0.60%**	**1207.613**	**138074.90**
GRTOP-S8	(5,5,0)	**59.318**	**1.000**	**3.40%**	**2226.280**	**200746.40**
GRTOP-S9	(7,3,0)	**60.726**	**1.000**	**5.20%**	**2157.578**	**173651.90**
GRTOP-S10	(1,9,0)	**63.999**	**1.000**	**12.30%**	**3256.597**	**201550.40**
avg.	-	**53.45**	**1.00**	**2.67%**	**1221.79**	**118147.08**
max	-	**63.99**	**1.020**	**12.30%**	**3256.597**	**201550.40**

4.5.3 Comparing the performance of different discretisation methods

In this section, we compare the performance of the numerical integration methods presented in Section 4.4. Table 4.3 summarises the results for the small range instances. The remaining columns refer to the aforementioned discretisation methods, namely the *Euler* method, Trapezoidal method (*TRP*), the third- and fourth-order Adams-Bashforth methods, $AB3$ and $AB4$, respectively, and both versions of the fourth-order Runge-Kutta method $1RK4$ and $2RK4$, where the former refers to the RK4 method with linear control interpolation of Equation (4.52) and the latter to the RK4 method with the interpolation described in Equation (4.53). Table 4.3 has been subdivided for each algorithm performance measure, namely, Status, UB, Error, Gap(%), CPU(s) and Tree, as in Table 4.2. Overall averages are shown at the end of each subdivision. The discretisation size and flight time horizon estimation have been kept as in the previous section.

From the results in Table 4.3, one can notice that the solver is more effective in proving optimality when the Euler and Trapezoidal methods used, finding 68 (75.6%) and 69 (76.7%) optimal solutions, respectively, against 62 (68.9%) and 61 (67.8%) optimal solutions found by using the 1RK4 and 2RK4 methods. Together, they also produce smaller

gaps for the instances that were not solved within the provided time limit. One can also verify that the ratio between the average number of branch-and-bound nodes and average CPU times is larger for these methods. This fact indicates that the Euler and Trapezoidal methods generate relaxations that are easier to solve during the tree search. On the other hand, Runge-Kutta methods outperform all the others in terms of error. The Runge-Kutta methods present error magnitudes that are up to 16 times smaller on average than the largest errors, at the expense of presenting higher average gaps for the instances that were not solved to optimality. There is a clear trade-off between computational performance and solution accuracy among the lower and higher order integration methods. Nonetheless, the third- and fourth-order Adams-Bashforth methods perform quite poorly for our problem, given the large error values and considerable gaps compared to the lower order methods.

We have extended our computational results for the smallest and largest instances of type "S" in Table 4.4. In this table, the first column shows the instance names and remaining columns present the error and CPU times for each discretisation methods presented in Section 4.4. The settings for the experiments shown in Table 4.4 remain the same as in the ones shown in Table 4.3.

From the results presented in Table 4.4, it can be seen that the magnitudes of the errors can be considered acceptable even for the lower order methods. To support our claim, we have further investigated which state variables are most affected by the errors when using the Euler's method. This has been accomplished by performing two modifications in our formulation. The first one consists of using a vector $\boldsymbol{\varepsilon} = (\varepsilon_x, \varepsilon_y, \varepsilon_h, \varepsilon_v, \varepsilon_\gamma, \varepsilon_\varphi), \boldsymbol{\varepsilon} \in \mathbb{R}^6$, to represent the error for each state variable in the dynamic equations. For example, the generic discretisation method presented in Constraints (4.22) and (4.23) can be re-written in the form of Constraints (4.61) and (4.62), respectively. It means that more variables will be added to the MISOCP formulation for the GRTOP.

$$\mathbf{y}_{g,t+1} \leq \mathbf{y}_{gt} + \eta f(\mathbf{y}_{gt}, \mathbf{u}_{gt}, t) + \boldsymbol{\varepsilon}, \forall g \in G, \forall t \in T \setminus \{N-1\} \tag{4.61}$$

$$\mathbf{y}_{g,t+1} \geq \mathbf{y}_{gt} + \eta f(\mathbf{y}_{gt}, \mathbf{u}_{gt}, t) - \boldsymbol{\varepsilon}, \forall g \in G, \forall t \in T \setminus \{N-1\} \tag{4.62}$$

The second modification follows from the first one as the objective function (4.8) needs to be re-written as in Equation (4.63). While the first term remains the same, the second term of the new objective function sums up the individual errors for each state variable.

$$\min \quad \sum_{g \in G} \sum_{i \in L} \sum_{t \in T} t b_{git} + \mathbf{1}^\top \boldsymbol{\varepsilon}. \tag{4.63}$$

Table 4.5 shows the results of this reformulation for a subset of instances (using Euler's method for discretising the dynamics). The main source of errors are the position components of the state vector. This can be explained by the fact that those components have the largest magnitude among all state variables, varying roughly between 0 and

TABLE 4.3: Summary of the results for different discretisation methods

Group	Euler	TRP	AB3	AB4	1RK4	2RK4
			Status			
GRTOP-S2	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)
GRTOP-S3	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)
GRTOP-S4	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)
GRTOP-S5	(9,1,0)	(10,0,0)	(10,0,0)	(9,1,0)	(8,2,0)	(7,3,0)
GRTOP-S6	(7,3,0)	(9,1,0)	(5,5,0)	(4,6,0)	(7,3,0)	(6,4,0)
GRTOP-S7	(9,1,0)	(8,2,0)	(5,5,0)	(3,7,0)	(6,4,0)	(6,4,0)
GRTOP-S8	(5,5,0)	(5,5,0)	(5,5,0)	(3,7,0)	(6,4,0)	(6,4,0)
GRTOP-S9	(7,3,0)	(6,4,0)	(2,8,0)	(1,9,0)	(3,7,0)	(3,7,0)
GRTOP-S10	(1,9,0)	(1,9,0)	(2,8,0)	(2,8,0)	(2,8,0)	(3,7,0)
			UB			
GRTOP-S2	32.500	32.391	33.918	34.608	**17.071**	17.147
GRTOP-S3	35.810	35.917	37.134	37.916	21.503	**21.426**
GRTOP-S4	50.073	49.980	52.991	54.308	25.370	**25.304**
GRTOP-S5	55.652	55.438	58.518	59.958	**33.812**	33.836
GRTOP-S6	60.125	60.041	65.205	67.312	**27.306**	27.513
GRTOP-S7	62.825	63.181	68.598	71.264	30.967	**30.029**
GRTOP-S8	59.318	59.264	64.882	69.578	**23.779**	23.834
GRTOP-S9	60.726	60.867	67.428	69.781	**28.886**	29.309
GRTOP-S10	63.999	64.742	69.735	73.615	**30.335**	30.514
avg.	53.45	53.54	57.60	59.82	26.56	**26.55**
max	64.00	64.74	69.74	73.62	**33.81**	33.84
			Error			
GRTOP-S2	6.076	6.071	6.040	6.079	**1.000**	1.077
GRTOP-S3	4.438	4.565	4.633	4.565	**1.000**	1.007
GRTOP-S4	6.825	6.963	7.134	7.207	**1.000**	1.045
GRTOP-S5	3.866	4.015	3.928	3.951	**1.000**	1.088
GRTOP-S6	9.525	9.454	9.518	9.789	**1.000**	1.309
GRTOP-S7	5.854	5.659	6.264	6.164	1.033	**1.000**
GRTOP-S8	14.396	14.608	15.477	16.025	1.028	**1.000**
GRTOP-S9	8.114	8.194	9.018	8.868	**1.000**	1.108
GRTOP-S10	12.478	13.235	12.937	12.141	**1.000**	1.382
avg.	7.952	8.085	8.328	8.310	**1.007**	1.113
max	14.40	14.61	15.48	16.02	**1.03**	1.38
			Gap(%)			
GRTOP-S2	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
GRTOP-S3	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
GRTOP-S4	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
GRTOP-S5	0.10%	**0.00%**	**0.00%**	0.90%	3.00%	0.40%
GRTOP-S6	2.40%	**0.50%**	4.10%	3.00%	6.50%	9.20%
GRTOP-S7	**0.60%**	1.30%	5.10%	9.50%	10.20%	3.90%
GRTOP-S8	3.40%	**3.20%**	6.20%	13.50%	9.80%	8.30%
GRTOP-S9	5.20%	**4.80%**	15.40%	12.30%	17.00%	15.60%
GRTOP-S10	**12.30%**	16.10%	16.50%	20.50%	34.20%	34.40%
avg.	**2.67%**	2.88%	5.26%	6.63%	8.97%	7.98%
max	**12.30%**	16.10%	16.50%	20.50%	34.20%	34.40%
			CPU(s)			
GRTOP-S2	7.12	9.09	13.62	20.55	6.96	**6.14**
GRTOP-S3	**14.02**	17.10	38.15	39.37	16.93	49.12
GRTOP-S4	**104.99**	151.17	338.46	714.07	306.09	530.00
GRTOP-S5	727.23	**146.84**	328.67	919.91	859.51	1282.27
GRTOP-S6	1294.72	**939.52**	2423.59	2778.01	1602.54	1878.44
GRTOP-S7	**1207.61**	1693.21	2517.73	3157.09	1952.83	1774.54
GRTOP-S8	2226.28	2159.28	2571.27	3071.74	**1813.62**	1958.09
GRTOP-S9	**2157.58**	2789.33	3276.83	3386.65	2959.64	2660.28
GRTOP-S10	3256.60	3277.13	3216.06	3274.96	3089.89	**2938.90**
avg.	**1221.79**	1242.52	1636.04	1929.15	1400.89	1453.09
max	3256.60	3277.13	3276.83	3386.65	3089.89	**2938.90**
			Tree			
GRTOP-S2	800.60	**491.50**	1536.60	2208.70	1349.80	605.00
GRTOP-S3	3250.30	**2772.60**	9364.90	6327.00	2937.30	13045.80
GRTOP-S4	**18475.50**	24006.80	47125.80	87958.70	74628.00	88431.30
GRTOP-S5	186538.20	**17292.70**	36597.40	85667.00	104206.00	197635.70
GRTOP-S6	140235.50	**88382.10**	165818.60	140593.00	114531.70	98954.10
GRTOP-S7	138074.90	157636.90	146485.20	133274.00	126049.60	**89096.20**
GRTOP-S8	200746.40	144065.70	122573.20	100995.50	95772.00	**95153.00**
GRTOP-S9	173651.90	174789.50	114985.30	**98945.70**	127546.60	100475.20
GRTOP-S10	201550.40	174476.00	110871.60	**82600.00**	112147.00	113462.60
avg.	**118147.08**	87101.53	83928.73	82063.29	84352.00	88539.88
max	**201550.40**	174789.50	165818.60	140593.00	127546.60	197635.70

1000. Even though the error associated to the translational dynamics is comparatively higher, they only represent a small fraction of the magnitudes of the position variables. For example, the error $\varepsilon_y = 11.58$ associated to the y variable for the small-ranged instance `grtopS_21_1` only represents 1.16% of the range of this state variable. The last two columns of Table 4.5 shows the results from the original formulation with a single error variable using Euler's method.

Finally, we analyse how the error and CPU times behave as the number of collocation points N increases. This results are shown in Figure 4.2. For this experiment, the

TABLE 4.4: Detailed results for error and CPU(s)

Name	Euler		TRP		AB3		AB4		1RK4		2RK4	
	error	CPU(s)	error	CPU(s)	error	CPU(s)	error	CPU(s)	error	CPU(s)	error	CPU(s)
grtopS_21_1	15.54	5.0	15.54	6.4	15.05	8.6	14.59	13.2	11.68	5.0	11.62	5.2
grtopS_21_2	18.63	6.4	16.57	8.5	15.46	12.4	16.53	19.0	1.83	5.3	1.66	4.7
grtopS_21_3	12.54	5.7	13.43	7.0	13.09	9.9	13.12	12.5	0.96	4.1	0.96	6.4
grtopS_21_4	12.02	3.7	12.96	6.6	12.74	8.4	13.76	12.1	0.50	3.9	0.56	3.0
grtopS_21_5	12.77	5.5	12.63	8.2	13.38	10.1	12.51	19.1	1.32	4.2	1.35	3.3
grtopS_22_1	12.19	7.9	14.23	13.3	12.95	23.8	12.83	22.8	0.68	11.0	1.48	12.8
grtopS_22_2	12.24	6.6	12.95	9.5	13.24	12.2	12.27	20.2	1.01	4.0	1.04	5.2
grtopS_22_3	13.07	11.5	13.12	12.5	14.64	23.9	13.72	41.3	3.55	23.6	4.54	9.9
grtopS_22_4	14.06	9.4	12.91	8.9	13.50	15.3	13.66	29.4	0.63	2.0	0.63	4.1
grtopS_22_5	14.95	9.6	13.58	10.0	13.15	11.6	15.10	15.9	0.56	6.5	0.63	7.0
avg.	**13.80**	**7.1**	**13.79**	**9.1**	**13.72**	**13.6**	**13.81**	**20.5**	**2.27**	**7.0**	**2.45**	**6.1**
grtopS_101_1	23.04	3601.6	26.40	3600.6	29.45	2603.9	25.66	2863.3	1.37	1732.4	1.24	2049.6
grtopS_101_2	23.17	3600.5	20.50	3600.4	20.78	3600.4	21.30	3600.3	1.24	3600.4	1.22	3600.4
grtopS_101_3	30.60	3600.5	25.93	3600.2	30.49	3600.5	23.68	3600.4	4.11	3600.5	3.50	3600.4
grtopS_101_4	23.39	160.3	23.27	367.2	24.80	753.3	24.36	1083.1	0.83	363.1	1.01	889.0
grtopS_101_5	24.94	3600.7	28.66	3600.5	26.31	3600.4	22.48	3600.4	2.22	3600.3	6.89	3600.3
grtopS_102_1	25.91	3600.5	22.28	3600.4	26.11	3600.5	24.67	3600.4	1.64	3600.3	0.78	3600.5
grtopS_102_2	24.60	3600.6	33.02	3600.6	27.00	3600.5	32.97	3600.4	2.85	3600.5	8.35	3600.7
grtopS_102_3	30.08	3600.5	26.27	3600.4	27.11	3600.2	25.29	3600.4	4.09	3600.4	1.69	1247.1
grtopS_102_4	22.01	3600.5	27.28	3600.5	21.67	3600.6	19.75	3600.4	1.07	3600.4	2.20	3600.6
grtopS_102_5	26.27	3600.3	35.81	3600.6	29.63	3600.3	26.99	3600.5	0.93	3600.6	1.26	3600.4
avg.	**25.40**	**3256.6**	**26.94**	**3277.1**	**26.33**	**3216.1**	**24.71**	**3275.0**	**2.04**	**3089.9**	**2.81**	**2938.9**

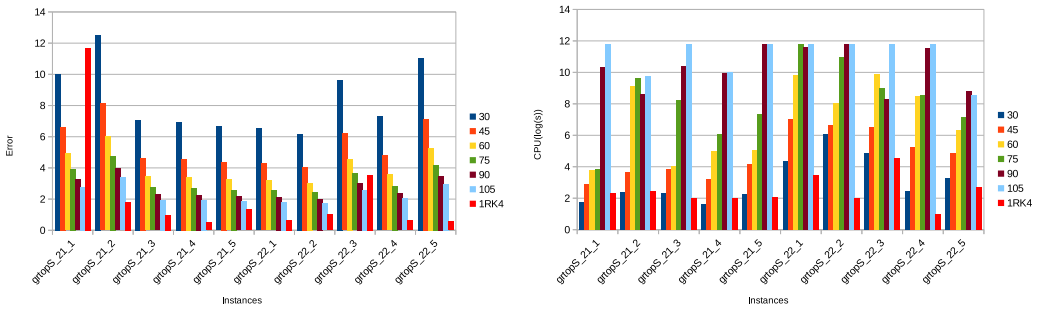
TABLE 4.5: Analysis of the error for the individual components of the state vector.

Name	x	y	h	v	γ	φ	Sum	CPU(s)	ε	CPU(s)
grtopS_21_1	0.00	11.58	0.00	0.00	0.68	0.00	12.26	6.34	15.54	4.98
grtopS_21_2	0.00	0.01	0.00	0.00	1.22	0.14	1.37	10.67	18.63	6.36
grtopS_21_3	0.00	0.79	0.00	0.00	1.14	0.00	1.93	6.58	12.54	5.70
grtopS_21_4	0.00	0.38	0.00	0.00	0.66	0.00	1.04	2.77	12.02	3.68
grtopS_21_5	0.00	0.00	0.00	0.00	0.99	0.00	0.99	3.77	12.77	5.54
grtopS_22_1	0.00	0.00	0.00	0.00	0.75	0.00	0.75	6.47	12.19	7.85
grtopS_22_2	0.00	0.00	0.00	0.00	0.89	0.00	0.89	5.01	12.24	6.59
grtopS_22_3	0.00	0.59	0.00	0.00	1.20	0.04	1.83	12.80	13.07	11.50
grtopS_22_4	0.00	0.00	0.00	0.00	0.66	0.00	0.66	1.47	14.06	9.42
grtopS_22_5	0.00	0.47	0.00	0.00	0.59	0.35	1.40	5.97	14.95	9.62

following number of collocation points have been adopted $N = \{30, 45, 60, 75, 90, 105\}$. The flight time interval has been computed as in the previous sections. Since the first term of the objective function (4.8) might affect the value of ε, we have eliminated this term from the objective function in order to properly assess the behaviour of the error term when varying the discretisation size. We highlight that these experiments were performed with the original MISOCP formulation (with a single error variable) and Euler's method. For the sake of comparison, we added a column representing the results from using the 1RK4 method with $N = 30$. Figure 4.2a shows the magnitude of the error for each number of collocation points on the small range instances of group GRTOP-S2 (expressed in the horizontal axis). One can notice that the error decreases as the number of collocation points increases, as expected. The opposite happens with the CPU times. In Figure 4.2b, these have been plotted in *log* scale.

4.5.4 Results for medium range instances

In this section, we present the results of our model for a subset of medium range instances. Here, we have chosen Euler's discretisation method due to its better average performance among the lower-order methods (Section 4.5.3). The number of collocation points N has been set to 60 and the flight time horizon τ_f has been chosen by the same procedure as described in Section 4.5.2.



(A) Magnitude of the error. (B) CPU(s) in log scale.

FIGURE 4.2: Behaviour of error and CPU times for several values of N.

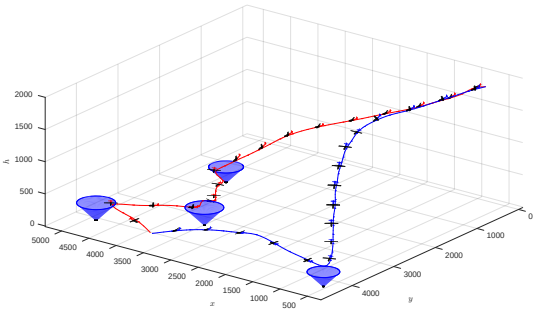
In Table 4.6, we show the results of our model for instances with $3, 4, 5$ and 6 waypoints. One will notice that running times for the instances solved to optimality have substantially increased compared to the small range instances, as well as the gap for the instances were optimality could not be proved. This happens because of the larger discretisation size that we have applied. Our choice on larger discretisation sizes seeks to guarantee the convergence of the integration methods. We also highlight that errors remain small compared to the range of the instances. The largest error for the results in Table 4.6 (89.83) represents only 1.8% of the magnitude of position variables related to this instance.

Figure 4.3 depicts the optimal solution for two instances, `grtopM_41_5` and `grtopM_42_3`. For illustration purposes, we have approximated the trajectories between collocation points by using splines. In the solution of instance `grtopM_41_5`, the flight times are 156s and 152s, for the first and second gliders, respectively, and the step size equals 4.34s. In the solution of instance `grtopM_42_3`, the flight times are 139s and 85s, for the first and second gliders, respectively, and the step size equals 3.39s. We have also provided video animations of feasible and optimal solutions of several instances as supplementary material and through the website Coutinho (2017).

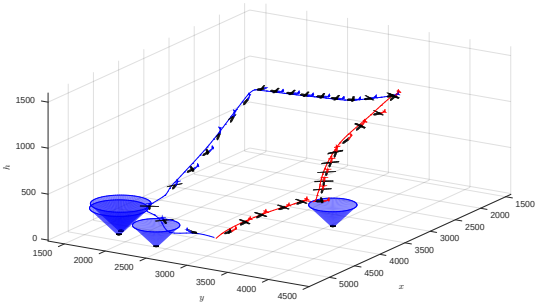
4.5.5 Routing and trajectory optimisation for disaster assessment

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



(A) Solution of `grtopM_41_5`.



(B) Solution of `grtopM_42_3`.

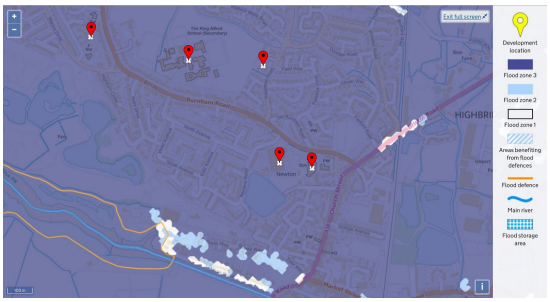
FIGURE 4.3: Depiction of the optimal solutions of two medium range instances.

TABLE 4.6: Results for a number of medium range instances

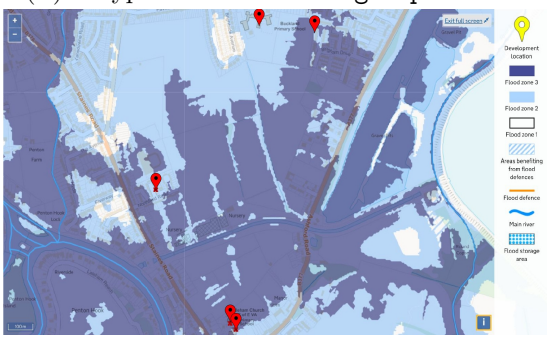
Name	Fleet	UB	LB	Error	CPU(s)	Tree	Gap(%)
grtopM_31_1	1	94.49	94.49	38.49	52.84	6559	0.00%
grtopM_31_2	1	83.43	83.43	28.43	99.77	10871	0.00%
grtopM_31_3	1	105.56	105.56	46.56	150.12	50459	0.00%
grtopM_31_4	1	135.85	135.85	76.85	35.71	4239	0.00%
grtopM_31_5	1	108.36	108.36	49.36	40.19	5323	0.00%
grtopM_32_1	1	73.59	73.59	24.59	167.19	15641	0.00%
grtopM_32_2	1	103.17	103.17	44.18	1098.67	262303	0.00%
grtopM_32_3	1	124.55	124.55	65.55	102.86	10618	0.00%
grtopM_32_4	1	95.88	75.81	37.88	3600.83	783809	21.00%
grtopM_32_5	1	91.79	91.79	33.80	260.57	38517	0.00%
grtopM_41_1	2	138.36	138.35	59.36	1943.31	115060	0.00%
grtopM_41_2	2	124.20	102.43	42.20	3600.47	185657	18.00%
grtopM_41_3	1	113.71	87.80	33.71	3600.79	334907	23.00%
grtopM_41_4	2	96.20	96.20	25.20	3080.92	306189	0.00%
grtopM_41_5	2	100.45	100.44	31.45	3174.82	192406	0.00%
grtopM_42_1	2	92.30	85.59	34.30	3600.32	122582	7.00%
grtopM_42_2	2	86.12	65.83	27.12	3600.28	99994	24.00%
grtopM_42_3	2	109.12	109.12	45.12	301.81	10467	0.00%
grtopM_42_4	2	153.41	97.07	71.41	3600.48	102879	37.00%
grtopM_42_5	1	129.94	129.94	54.94	2376.52	177920	0.00%
grtopM_51_1	2	160.34	99.19	82.34	3600.70	272174	38.00%
grtopM_51_2	2	122.69	107.50	43.69	3600.44	146891	12.00%
grtopM_51_3	2	154.80	146.37	72.80	3600.26	167945	5.00%
grtopM_51_4	2	116.08	105.20	28.08	3600.47	250171	9.00%
grtopM_51_5	2	137.60	88.60	58.60	3600.51	293565	36.00%
grtopM_52_1	2	148.99	148.99	59.99	1083.69	26234	0.00%
grtopM_52_2	2	93.96	79.15	41.96	3600.53	168447	16.00%
grtopM_52_3	2	119.81	85.83	36.81	3600.53	151592	28.00%
grtopM_52_4	2	141.57	95.62	51.57	3600.67	147272	32.00%
grtopM_52_5	2	159.38	91.63	75.38	3600.77	236883	43.00%
grtopM_61_1	3	168.25	107.26	37.25	3600.59	121569	36.00%
grtopM_61_2	3	126.14	112.61	49.14	3600.48	136656	11.00%
grtopM_61_3	3	182.83	79.91	89.83	3600.58	96375	56.00%
grtopM_61_4	2	159.76	112.63	73.76	3600.43	149115	30.00%
grtopM_61_5	2	121.34	80.89	43.34	3600.54	105711	33.00%
grtopM_62_1	2	141.84	81.54	50.84	3600.46	82085	43.00%
grtopM_62_2	2	167.52	78.92	88.52	3600.47	84507	53.00%
grtopM_62_3	3	193.56	154.27	78.56	3600.44	40665	20.00%
grtopM_62_4	3	163.16	78.92	40.16	3600.65	108157	52.00%
grtopM_62_5	2	153.53	91.17	50.53	3600.61	79393	41.00%

sites has been chosen in order to match the dimensions of the real locations. Figure 4.4 depicts the maps indicating the selection of waypoints for two flood-prone areas of two

cities in the UK, namely, London and Highbridge.



(A) Waypoints selection for `grtopS_lond1`.



(B) Waypoints selection for `grtopS_highb`.

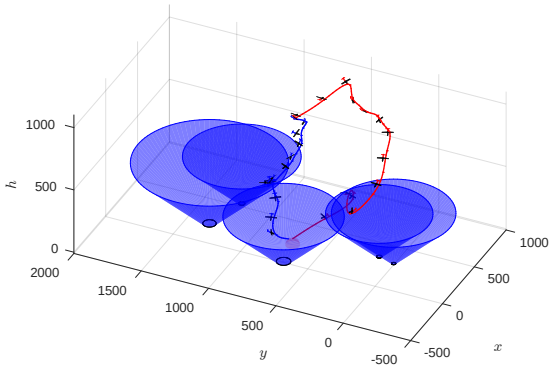
FIGURE 4.4: Maps of flood-prone areas in London and Highbridge. Source: Government Digital Service (2017).

Table 4.7 shows the results of our experiments with the generated UK instances. In this table, column $\#W$ represents the number of waypoints in the instance. The remaining columns keep the same meaning as in Table 4.6. The number of landing sites for all instances was set to 1. The number of collocation points has been set to $N = 30$. Figure 4.5 depicts the solutions for 2 UK instances, namely, `grtopS_lond1` (London) and `grtopS_highb` (Highbridge).

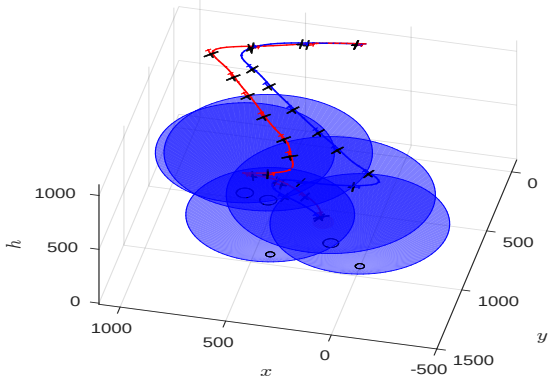
TABLE 4.7: Results of the GRTOP formulation for the UK instances

Name	#W	Fleet	UB	LB	Error	CPU(s)	Tree	Gap(%)
grtop_bost1	7	3	73.03	73.03	34.03	263.95	9973	0.00%
grtop_bost2	7	2	75.38	75.15	31.38	3600.16	391381	0.00%
grtop_bost3	7	3	69.07	69.07	31.07	255.58	24063	0.00%
grtop_highb	5	2	57.08	57.08	23.08	34.23	830	0.00%
grtop_lond1	5	2	59.44	59.44	27.44	54.46	4923	0.00%
grtop_lond2	7	3	73.86	73.86	34.86	106.89	3059	0.00%
grtop_lond3	10	1	77.41	75.09	34.41	3600.42	241168	3.00%
grtop_lond4	7	3	75.48	75.48	30.48	142.17	5207	0.00%
grtop_moore	7	1	80.39	80.39	25.40	51.51	751	0.00%
grtop_prtsm	5	1	65.59	65.59	28.59	103.49	16355	0.00%

From the results in Table 4.7, one can observe that all instances are solved to optimality, except `grtop_lond3`. The results of numerical experiments with the UK instances are similar to the results achieved for the small-ranged random instances.



(A) Solution of `grtopS_lond1`.



(B) Solution of `grtopS_highb`.

FIGURE 4.5: Depiction of the optimal solutions of two UK instances.

4.6 Conclusions of the chapter

In this chapter, we have considered the GRTOP. This problem has been motivated by a disaster assessment application. In the GRTOP, we are asked to simultaneously find optimal routes and trajectories for a fleet of unmanned aerial gliders. The fleet of gliders is modelled by their EOMs, which consist of a set of ordinary differential equations. We propose a novel MINLP formulation for the GRTOP, which accounts for routing and trajectory optimisation in an integrated framework. In order to avoid a non-convex formulation, we linearise the gliders' EOM using a set of steady-state conditions. This reduces the MINLP to a MISOCP problem. To allow for a more tractable problem while keeping the quality of our solutions, we relax the linear dynamic equations and penalise the corresponding error term in the objective function. We present several discretisation methods for the resulting linear dynamic equations.

In order to test our model, we have generated 180 random instances. We compared different commercial solvers on a subset of instances, namely, CPLEX, Gurobi and Xpress. Based on the results, Xpress was chosen for the next experiments. The second set of experiments was concerned about the discretisation methods and errors. Higher order integration methods were able of achieving smaller error magnitudes, but at the expense of CPU times. On the other hand, lower order methods typically reduced computation times, at the expense of solution accuracy. A detailed analysis has been

carried out regarding the magnitude of the error. It was shown that the errors represent a small fraction of the magnitudes of the state variables and therefore are considered acceptable. Experiments on a subset of instances showed that the error is mostly due to the position variables, which have higher magnitude than the variables regarding angular orientation and airspeed. Finally, we studied the effect of increasing the number of collocation points on the magnitudes of error and CPU times. In general, the trade-off between error magnitudes and CPU times becomes clear on the choice of N.

The results for medium range instances showed that acceptable accuracy, i.e., negligible error magnitudes compared to the magnitudes of the state variables, can be achieved for long range flights even by employing lower-order discretisation methods. These results are confirmed in Section 4.5.3. In order to guarantee the convergence of integration methods, the number of collocation points has to be increased. This has a direct influence on the number of medium range instances solved to optimality. Finally, we present results for the so-called UK instances. These instances are created for disaster assessment in UK cities with high flooding risk. The model presents a good performance over instances from this group.

The accuracy of our solutions could be further improved by applying sequential linearisation at each node of the B&B tree, but this would dramatically increase the computation times. Further research could also focus on an adaptive integration method instead of the fixed time grid we have used. However, this improvements would also increase the CPU times.

Alternative objective functions can also be tested. For example, in disaster assessment applications, one might prefer to minimise the duration of the longest route (a.k.a. makespan) rather than the total flight time. A second scenario could consider, for example, minimising the latency, which is related to the arrival times at each waypoint.

Our formulation is capable to tackle a good number of test cases. However, for the instances with a larger number of waypoints and for higher discretisation sizes, we were unable to prove optimality. This motivates the development of heuristic methods that should be able to find good solutions in small CPU times.

Chapter 5

A Trajectory Optimisation-based Matheuristic for the GRTOP

In this chapter, we present a matheuristic algorithm for solving the GRTOP. The proposed algorithm is based on two main building blocks: (i) a STO heuristic; and an ILS -based matheuristic. A background on the ILS metaheuristic is provided before we discuss the proposed algorithm for the GRTOP.

5.1 Background on the ILS algorithm

Many exact algorithms have been proposed for solving several variants of VRPs, see, for example, the Capacitated VRP (Pecin et al., 2017), the Green VRP (Andelmin and Bartolini, 2017) and the Time Dependent TSP in Controlled Airspace (Furini et al., 2016). However, heuristic methods are still preferred for practical applications due their advantage in terms of solution quality vs. computational time. In fact, the survey by Braekers et al. (2016) shows that, between 2009 and 2015, around of 80% of the papers on VRPs presented heuristic algorithms as a solution method. We refer the interested reader to the book by Toth and Vigo (2002), for instance, for an overview of both exact and heuristic algorithms for VRPs.

Heuristic methods for VRPs are usually divided into two distinct classes, namely, classical heuristics and metaheuristics. Classical heuristics, in general, are formed by constructive, two phase and improvement heuristics. In turn, metaheuristics are higher-level procedures designed to manage a set of lower-level heuristics in order to solve complex optimisation problems. One of the most employed metaheuristics in the literature for solving VRPs is the Iterated Local Search (ILS) algorithm. See, for example, Subramanian et al. (2013), Silva et al. (2015), Penna et al. (2017) and Xie et al. (2017).

An ILS metaheuristic is composed by four basic components. The first one consists of an algorithm for generating initial solutions, here referred to as `GenerateInitialSolution`. This step is usually implemented as a set of fast and simple algorithms for finding a feasible solution in short computing times. The second component consists of a `LocalSearch` procedure, where the current solution is refined into a better one in terms of objective function value. For *intensification* purposes, the `LocalSearch` step is often implemented on a restricted solution space rather than on the entire solution space. The third component is a perturbation mechanism (here called `Perturbation`), where the current solution is modified in order to escape a local optimum solution (this is usually known in the literature as a *diversification* step). The last component is the `AcceptanceCriterion`, which defines from which solution the method should continue. Figure 5.1 illustrates how an ILS algorithm explores the solution space. Let us define the point a in Figure 5.1 as the starting point returned by the `GenerateInitialSolution` method. A local search is performed on the neighbourhood of a and the local optimum solution b is found. Note that continuing the search around the neighbourhood of b is redundant, therefore the `Perturbation` procedure is applied so that an unexplored neighbourhood can be searched. In Figure 5.1, the `Perturbation` mechanism leads the current solution from point b to point c, where a new local search can be performed and a better local optimum d can be found. The method continues until a stopping criteria is met. A popular choice is the maximum number of iterations with no improvement of the current best solution.

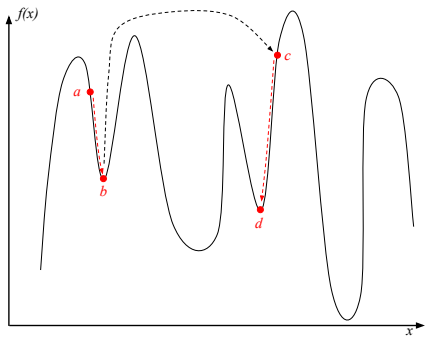


FIGURE 5.1: Illustration of the ILS procedure. The red dashed line represents the local search step. The black dashed line represents the perturbation step.

Other heuristic and metaheuristic frameworks have also been applied to solve several variants of VRPs with success. A few examples include, Ropke and Pisinger (2006), that proposed a large neighbourhood search for solving several classes of VRPs with backhauls, and Vidal et al. (2012) that proposed a hybrid genetic algorithm for multi-depot and periodic VRPs. However, as discussed before, we adopt an ILS method due to its simplicity and robustness on solving a variety of VRPs (Silva et al., 2012; Penna et al., 2013; Subramanian and Battarra, 2013; Martinelli et al., 2013; Vidal et al., 2015; Kramer et al., 2015; Cruz et al., 2017; Bulhões et al., 2017).

5.2 Additional equilibrium flight modes

In this chapter, we employ the same glider dynamics as presented in Chapter 3. Let us recall $\mathbf{y}_{eq} = (x_{eq}, y_{eq}, h_{eq}, v_{eq}, \gamma_{eq}, \varphi_{eq})$ and $\mathbf{u}_{eq} = (C_{L,eq}, \mu_{eq})$ as the steady-states and controls, respectively, of the dynamics defined by Equation (3.13). Steady-flight conditions are defined such that the derivatives of the state variable with respect to time are zero, that is:

$$\dot{\mathbf{y}} = f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau) = \mathbf{0} \in \mathbb{R}^6. \tag{5.1}$$

In this chapter, we are interested in finding equilibrium flight conditions for two distinct situations, namely, steady-level flight, in which the origin and destination points are nearly in the same altitude, and steady-descent flight, if the origin is in a higher altitude and the glider must descent in order to reach the desired destination.

5.2.1 Steady-level flight conditions

In Section 4.4.1, under certain simplifying assumptions, we apply a set of analytical steady-state conditions for a gliding level-flight as described, e.g., in Russell (1996). In this chapter, we use a numerical approach instead, which is based on the method presented by (Stengel, 2004, p. 257) for computing more accurate steady level-flight conditions. However, our formulation differs from the one presented by Stengel (2004) by the addition of box constraints on state and control variables. Let $\mathbf{y}^*$ and $\mathbf{u}^*$ be the optimal solution of the optimisation problem defined by Equations (5.2–5.4) for an arbitrary time instant τ. We approximate the steady-states of Equation (4.32) by $\mathbf{y}^* \in \mathbb{R}^6$ and $\mathbf{u}^* \in \mathbb{R}^2$, i.e., $\mathbf{y}_{eq} = \mathbf{y}^*$ and $\mathbf{u}_{eq} = \mathbf{u}^*$.

$$\min_{\mathbf{y},\mathbf{u}} \quad ||f(\mathbf{y}, \mathbf{u})||_2 \tag{5.2}$$

$$s.t. \quad \mathbf{y}_{lb} \leq \mathbf{y} \leq \mathbf{y}_{ub} \tag{5.3}$$

$$\mathbf{u}_{lb} \leq \mathbf{u} \leq \mathbf{u}_{ub} \tag{5.4}$$

In the problem defined by Equations (5.2–5.4), the objective function (5.2) minimises the Euclidean norm of the right hand side of Equation (5.1) for an arbitrary τ. By minimising such norm, we expect to find a solution that fulfils the condition in Equation (5.1), if such a solution exists. Otherwise, an approximative solution is returned. Constraints (5.3) and (5.4) ensure that the optimal steady-flight conditions lie within the bounds on state and control variables.

In Table 5.1, we compare the steady-states presented in Section 4.4.1 with the ones found by using the well known MATLAB's function `fmincon` to solve the problem defined by

Equations (5.2–5.4). We refer to these steady-states as s_1 and s_2, respectively. We fixed the reference altitude to 500 metres to match the parameter for the wind strength as defined in Section 3.3. Table 5.1 shows that s_2 (numerically generated) is closer to fulfilling the condition defined in Equation (4.32) than s_1 (generated analytically). Therefore, for the remainder of this chapter, we will use s_2 as level flight steady-state conditions.

TABLE 5.1: Comparison of steady-level flight states.

	h_{eq}	v_{eq}	γ_{eq}	φ_{eq}	$C_{L,eq}$	μ_{eq}	$\|\|f(\mathbf{y},\mathbf{u})\|\|_2$
s_1	500.00	9.45	-0.04	0.0	0.74	0.0	15.6705
s_2	500.00	12.48	-0.02	-1.57	0.37	0.0	0.4133

5.2.2 Steady-descent flight

Computing steady-descent flight conditions numerically for every pair of points would require a considerable computational effort. Therefore, in this section, we resort to an analytical approach for finding descent-flight equilibrium states and controls. The forces acting on a UAV in steady-descent flight are the *lift* (L), drag (D) and weight (defined as the mass of the glider m_g times the gravity on earth g_e). In a descent gliding flight, these forces are on equilibrium and can be depicted as in Figure 5.2. It is assumed that the equilibrium roll and yaw angles are close to zero, i.e., $\mu_{eq} \approx \varphi_{eq} \approx 0$. If we approximate the equilibrium flight path angle (γ_{eq}) by the angle between points (x_1, y_1, h_1) and (x_2, y_2, h_2) compared to a horizontal plane, then γ_{eq} can be written as in Equation (5.5), where R is the flight range and $\Delta h = h_2 - h_1$.

$$\gamma_{eq} \approx -\tan^{-1}(\Delta h / R) \tag{5.5}$$

From the triangle of forces in Figure 5.2, one can write an expression relating γ_{eq} with lift and drag forces as in Equation (5.6). The rightmost expression in Equation (5.6) follows from the definition of the lift and drag coefficients, respectively, C_L and C_D (Russell, 1996, p. 42).

$$\cos\gamma_{eq} = \frac{L}{\sqrt{L^2 + D^2}} = \frac{C_L}{\sqrt{C_L^2 + C_D^2}} \tag{5.6}$$

Solving Equation (5.6) for C_L leads to a quadratic equation with two possible solutions. The two roots define a minimum and a maximum *angle of attack* (Stengel, 2004, p. 53). By choosing the minimum C_L (Equation (5.7)) we indirectly choose the smallest angle of attack and therefore the smallest amount of generated lift, thus allowing the glider to perform a steady-descent flight. In Equation (5.7), let $b = 1 + \sqrt{2C_{D0}k_A} - \cos\gamma_{eq}^{-1}$ and $\Delta = b^2 - 4k_A C_{D0}$, where the glider parameters C_{D0} (coefficient of drag at zero-lift) and

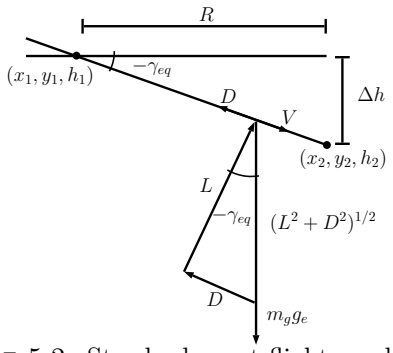


FIGURE 5.2: Steady descent flight conditions.

k_A (aerodynamic coefficient) are defined in Section 3.3. Here we denote by $\Re(.)$ as the real part of a complex number.

$$C_{L,eq} = \min \left\{ \frac{-b \pm \Re(\sqrt{\Delta})}{2k_A} \right\} \tag{5.7}$$

Finally, from the lift equation (Russell, 1996, p. 42), one can write an expression for the equilibrium velocity as in Equation (5.8), where the glider parameters m_g (the mass of the glider) and S (wing area), and the density of the air (ρ) are defined as in Chapter 3.

$$v_{eq} = \begin{cases} \sqrt{\frac{2m_g g_e \cos\gamma_{eq}}{\rho S_w C_{L,eq}}}, \text{ if } C_{L,eq} > 0 \\ v_{lb}, \text{ otherwise.} \end{cases} \tag{5.8}$$

By approximating the steady-descent flight path (γ_{eq}), lift coefficient ($C_{L,eq}$) and velocity (v_{eq}) using Equations (5.5), (5.7) and (5.8), respectively, one might en up with equilibrium values that are out of the bounds defined for state and control variables. In this case, we simply round the respective equilibrium value to the nearest bound.

5.3 A multi-phase mixed-integer trajectory optimisation formulation for the GRTOP

The problem defined in this chapter closely follows the definition in Chapter 4. However, we introduce two main modifications. Firstly, we treat the TO as a multi-phase problem, this allows for changes to the system dynamics during the flight to be taken into account. Secondly, instead of minimising the total mission duration we seek to minimise the maximum route duration, i.e., the so-called makespan.

We associate a phase of the TO to each arc $(i, j) \in A$. In this work, we use the words phase and arc interchangeably. Let us define $\mathbf{y}_{ijg}(\tau_{ijg}) : [0, \infty) \to \mathbb{R}^6$ and $\mathbf{u}_{ijg}(\tau_{ijg}) : [0, \infty) \to \mathbb{R}^2$ 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^o_{ijg} and τ^f_{ijg} represent the initial

and final flight times of the glider g flying on arc (i,j) such that $\tau_{ijg} \in [\tau^o_{ijg}, \tau^f_{ijg}]$. We recall that the unknown states and controls are interpreted as the evolution of the dynamical system (Equation 3.13), where τ is the independent variable.

The EOMs of the glider g flying on arc (i,j) can be expressed as in Equation (5.9). Both state and control variables are limited by lower and upper bounds, as explained in Section 4.3.1. Initial conditions $\mathbf{y}_o$ and $\mathbf{u}_o$ must be provided at time 0, i.e., $\mathbf{y}_{ijg}(0) = \mathbf{y}_o$ and $\mathbf{u}_{ijg}(0) = \mathbf{u}_o, \forall (i,j) \in A$ and $\forall g \in G$.

$$\dot{\mathbf{y}}_{ijg} = \mathbf{f}_g(\mathbf{y}_{ijg}(\tau_{ijg}), \mathbf{u}_{ijg}(\tau_{ijg}), \tau_{ijg}) \tag{5.9}$$

Figure 5.3 illustrates our conceptual graph construction for a small example and its respective feasible solution. In our example, let the launching point be 0, $V = \{1,2\}$, $L = \{3\}$, and $G = \{1,2\}$. For each arc in the set $A = \{(0,1),(0,2),(1,2),(2,1),(1,3),(2,3)\}$ there are associated time τ_{ijg}, state $\mathbf{y}_{ijg}(\tau_{ijg})$ and control $\mathbf{u}_{ijg}(\tau_{ijg})$ variables, as well as variable time limits τ^o_{ijg} and τ^f_{ijg} and a dynamical system as in Equation (5.9), where $(i,j) \in A$ and $g \in G$. In Figure 5.3, the flight duration of the gliders 1 and 2 can be computed as $\Delta t_1 = (\tau^f_{011} - \tau^o_{011}) + (\tau^f_{131} - \tau^o_{131})$ and $\Delta t_2 = (\tau^f_{022} - \tau^o_{022}) + (\tau^f_{232} - \tau^o_{232})$. Note that, in Figure 5.3, the continuity of dynamics for each glider is enforced by setting the initial flight time (and respective state and control variables) of an arc equal to the final flight time in the previous arc in the same route.

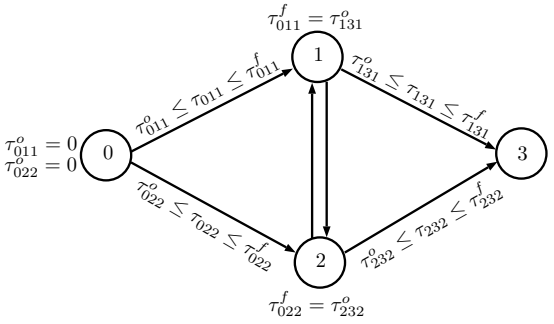


FIGURE 5.3: A small example graph representing routes $r_1 = (0,1,3)$ e $r_2 = (0,2,3)$ and their respective time, state and control continuity constraints.

Now 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., $A' = A \setminus \{(0,j), j \in V\}$. Thus, the objective function (5.10) minimises the flight duration of the longest route.

$$\min \max_{g \in G} \left\{ \sum_{(i,j) \in A} (\tau^f_{ijg} - \tau^o_{ijg}) a_{ijg} \right\} \tag{5.10}$$

Constraints (5.11–5.13) account for the assignment of routes to gliders. Constraint (5.11) ensures that every used glider lands in one of the predetermined landing sites.

Constraints (5.12) and (5.13) make sure that every waypoint is visited at least once and that the continuity of routes is preserved.

$$\sum_{i \in V} a_{0ig} = \sum_{i \in V} a_{ilg} \leq 1, \forall l \in L, \forall g \in G \tag{5.11}$$

$$\sum_{g \in G} \sum_{\substack{i \in V \\ i \neq j}} a_{ijg} \geq 1, \forall j \in V \tag{5.12}$$

$$\sum_{\substack{i \in V \\ i \neq j}} a_{ijg} - \sum_{\substack{i \in V \\ i \neq j}} a_{jig} = 0, \forall j \in V, \forall g \in G \tag{5.13}$$

Constraints (5.14–5.16) ensure that the gliders lie within the respective covering regions of the waypoints and landing sites at the end points of each phase. A phase at an arc (i, j) is deemed to start and end at its respective initial and final flight times τ^o_{ijg} and τ^f_{ijg}. Constraints (5.14) and (5.15) ensure that a glider g is flies within the boundaries of waypoint i at time τ^o_{ijg}, if arc (i, j) is used. While Constraints (5.16) state that a glider g must touch or go inside a landing site at the end of its mission.

$$a_{ijg}((x_{ijg}(\tau^o_{ijg}) - \bar{x}_i)^2 + (y_{ijg}(\tau^o_{ijg}) - \bar{y}_i)^2) \leq (h_{ijg}(\tau^o_{ijg}) + \bar{r}_i)^2, \forall (i, j) \in A', \forall g \in G \tag{5.14}$$

$$a_{ijg}\underline{h}_i \leq a_{ijg}h_{ijg}(\tau^o_{ijg}) \leq \bar{h}_i, \forall (i, j) \in A', \forall g \in G \tag{5.15}$$

$$a_{ijg}((x_{ijg}(\tau^f_{ijg}) - \tilde{x}_i)^2 + (y_{ijg}(\tau^f_{ijg}) - \tilde{y}_i)^2 + h^2_{ijg}(\tau^f_{ijg})) \leq \tilde{r}^2_j, \forall i \in V, \forall j \in L, \forall g \in G \tag{5.16}$$

Having the gliders at an unfavourable angular orientation (a.k.a. attitude) when taking pictures is an undesirable solution that must be avoided. Constraints (5.17) and (5.18) make sure that glider g is nearly in a "flat" position (i.e., in level flight within very small flight path ($\hat{\gamma}$) and roll ($\hat{\mu}$) angles) at the moment it takes a picture of waypoint i.

$$-\hat{\gamma} \leq a_{ijg}\gamma_{ijg}(\tau^o_{ijg}) \leq \hat{\gamma}, \forall (i, j) \in A', \forall g \in G \tag{5.17}$$

$$-\hat{\mu} \leq a_{ijg}\mu_{ijg}(\tau^o_{ijg}) \leq \hat{\mu}, \forall (i, j) \in A', \forall g \in G \tag{5.18}$$

Constraints (5.19–5.23) are related to the flight dynamics of the gliders. Constraints (5.19) enforce the flight dynamics of glider g to be applied if this glider flies through arc (i, j). Constraints (5.20–5.22) ensure the continuity of state, control and time variables is maintained if arc (i, j) precedes arc (j, k) in a solution. Constraints (5.23) keep the time variable within its bounds on arc (i, j).

$$\dot{\mathbf{y}}_{ijg} = \mathbf{f}_g(\mathbf{y}_{ijg}(\tau_{ijg}), \mathbf{u}_{ijg}(\tau_{ijg}), \tau_{ijg})a_{ijg}, \forall (i, j) \in A, \forall g \in G \tag{5.19}$$

$$\mathbf{y}_{jkg}(\tau^o_{jkg})a_{ijg} = \mathbf{y}_{ijg}(\tau^f_{ijg})a_{ijg}a_{jkg}, \forall (i, j), (j, k) \in A, \forall g \in G \tag{5.20}$$

$$\mathbf{u}_{jkg}(\tau^o_{jkg})a_{ijg} = \mathbf{u}_{ijg}(\tau^f_{ijg})a_{ijg}a_{jkg}, \forall (i, j), (j, k) \in A, \forall g \in G \tag{5.21}$$

$$\tau^o_{jkg} a_{ijg} = \tau^f_{ijg} a_{ijg} a_{jkg}, \forall (i,j),(j,k) \in A, \forall g \in G \tag{5.22}$$

$$\tau^o_{ijg} \leq \tau_{ijg} \leq \tau^f_{ijg}, \forall (i,j) \in A, \forall g \in G \tag{5.23}$$

Finally, Expressions (5.24–5.27) define the domain of the optimisation variables.

$$a_{ijg} \in \{0,1\}, \forall (i,j) \in A, \forall g \in G \tag{5.24}$$

$$\mathbf{y}_{ijg}(\tau_{ijg}) \in \mathbb{R}^6, \forall (i,j) \in A, \forall g \in G \tag{5.25}$$

$$\mathbf{u}_{ijg}(\tau_{ijg}) \in \mathbb{R}^2, \forall (i,j) \in A, \forall g \in G \tag{5.26}$$

$$\tau^o_{ijg}, \tau^f_{ijg} \in \mathbb{R}. \forall (i,j) \in A, \forall g \in G \tag{5.27}$$

5.4 A heuristic method for trajectory optimisation

In this section, we propose an efficient trajectory optimisation heuristic developed for finding feasible trajectories for a glider flying through several waypoints. Our so-called STO heuristic can be summarised as follows. In order to simplify the glider's dynamics we linearise the EOMs (3.13) under the steady-state conditions presented in Section 5.2. Next, for a given glider and its assigned route (sequence of waypoints), we divide the glider's multi-phase trajectory flying through the waypoints into several phases, one for each arc (or segment) of the route. Each phase can be solved efficiently, for a fixed flight duration, by a SOCP TO formulation. The flight time on each arc of the route is minimised heuristically by using the SOCP formulation as a subproblem. In the next sections, we discuss each component of our algorithm in details.

In order to simplify Equation (3.13) into a more numerically tractable form, we employ the same linearisation technique used in Section 4.4.1. Let us recall variables $\delta\mathbf{y}(\tau) = \mathbf{y}(\tau) - \mathbf{y}_{eq}$ and $\delta\mathbf{u}(\tau) = \mathbf{u}(\tau) - \mathbf{u}_{eq}$ as perturbations over state and control variables, respectively. A linear dynamics can be achieved by applying the first order Taylor's expansion to Equation (3.13):

$$T(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \delta\mathbf{y}, \delta\mathbf{u}, \tau) = f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau) + \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{y}} \delta\mathbf{y}(\tau) + \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{u}} \delta\mathbf{u}(\tau). \tag{5.28}$$

When considering the the equilibrium condition, the first term of Equation (5.28) equals zero by definition. Moreover, we disregard he higher order terms of the Taylor's expansion for convenience. Matrices $A = \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{y}}$ and $B = \frac{\partial f(\mathbf{y}_{eq}, \mathbf{u}_{eq}, \tau)}{\partial \mathbf{u}}$ represent the Jacobians of the EOMs (3.13) with respect to state and control variables. Let us define $\mathbf{y}_{eq(ij)}$ and $\mathbf{u}_{eq(ij)}$ as the equilibrium state and controls of arc $(i,j) \in A$. Hence, we can approximate the EOMs (5.9) of the glider g flying through arc (i,j) by the linear system

dynamics in state-space form as shown in Equation (5.29).

$$\dot{\mathbf{y}}_{ijg} = A\delta\mathbf{y}_{ijg}(\tau) + B\delta\mathbf{u}_{ijg}(\tau), \tag{5.29}$$

The linear EOMs (5.29) are expected to be a good approximation of Equation (3.13), provided that the glider is restricted to small motions around the equilibrium conditions $\mathbf{y}_{eq}$ and $\mathbf{u}_{eq}$. A simple numerical integration experiment can be carried out to show that by constraining the control variables $\mathbf{u}$ to small perturbations around $\mathbf{u}_{eq}$ leads to a satisfactory approximation of the actual system dynamics. In fact, our preliminary computational experiments showed that adding small perturbation constraints for $\mathbf{u}$ into our optimisation framework was already sufficient to decrease approximation errors.

5.4.1 A single phase trajectory optimisation subproblem

In this section, we are interested in finding an optimal trajectory for a glider g flying through an arc $(i, j) \in A$, which turns out to be a subproblem in our STO heuristic. Such optimal trajectory is determined by solving a SOCP formulation presented further in this section. For simplicity, we will omit the i, j and g subscripts on state, control and time variables. It is assumed a fixed flight duration, denoted as $\Delta\tau = \tau^f - \tau^o$, for the glider g flying on arc (i, j).

In this section, we employ a direct collocation method for optimising the trajectory of a glider flying through a single arc of a route. Let $\mathbf{y}(\tau)$ and $\mathbf{u}(\tau)$ be the state and control variables of a glider g flying through arc (i, j), where $\tau \in [\tau^o, \tau^f]$, and let $T = \{0, \ldots, N-1\}$ be the set of N *collocation points* (or time indices). By discretising the time interval $[\tau^o, \tau^f]$ over T, we can define a uniform time grid $\tau_t = \tau_o + \eta t$, where the index $t \in T$ represents a time instant τ_t within the interval $[\tau^o, \tau^f]$ and $\eta = \frac{\tau_f - \tau_o}{N-1}$ is an uniform step size. Let $\mathbf{y}_t$ and $\mathbf{u}_t$ represent the approximations of the state and control, respectively, at time τ_t. Computational experiments in Section 4.5.3 suggested that Euler's discretisation usually yields simpler optimisation problems that can be solved efficiently by existing solvers. The Euler's method applied to the linear dynamical system defined by Equation (5.29) leads to the following discretised EOMs, with predefined initial conditions $\mathbf{y}^o$ and $\mathbf{u}^o$:

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \eta(A\delta\mathbf{y}_t + B\delta\mathbf{u}_t), \forall t \in T. \tag{5.30}$$

$$\mathbf{y}_0 = \mathbf{y}^o, \mathbf{u}_0 = \mathbf{u}^o \tag{5.31}$$

Equations (5.30–5.31) will be embedded into our TO subproblem as part of our direct collocation procedure. The feasibility of the EOM (5.30) can be improved by relaxing

its equality requirement around an error variable, here denoted as ε. Constraints (5.32–5.33) will substitute Constraints (5.30) in our subproblem formulation.

$$\mathbf{y}_{t+1} \leq \mathbf{y}_t + \eta(A\delta\mathbf{y}_t + B\delta\mathbf{u}_t) + \mathbf{1}\varepsilon \tag{5.32}$$

$$\mathbf{y}_{t+1} \geq \mathbf{y}_t + \eta(A\delta\mathbf{y}_t + B\delta\mathbf{u}_t) - \mathbf{1}\varepsilon. \tag{5.33}$$

By assuming that waypoints are visited in the last time step, here denoted as t^f, one can define the visiting constraints for the glider flying to any waypoint in set V by means of Inequalities (5.34–5.37). More precisely, Constraints (5.34) and (5.35) ensure that the glider will lie within the waypoint's boundaries at the last time step corresponding to the phase (i, j) whereas Constraints (5.36) and (5.37) guarantee that the glider will be in a proper angular orientation in order to take a picture, i.e., the final flight path and roll angles should be within a small interval predetermined, respectively, by $\hat{\gamma}$ and $\hat{\mu}$.

$$(x_{t_f} - \bar{x})^2 + (y_{t_f} - \bar{y})^2 \leq (h_{t_f} + \bar{r})^2 \tag{5.34}$$

$$\underline{h} \leq h_{t_f} \leq \bar{h} \tag{5.35}$$

$$-\hat{\gamma} \leq \gamma_{t_f} \leq \hat{\gamma} \tag{5.36}$$

$$-\hat{\mu} \leq \mu_{t_f} \leq \hat{\mu} \tag{5.37}$$

The constraint regarding landing sites can be defined in a similar way, i.e., for any arc $(i, j), j \in L$, Constraint (5.38) ensures that the glider is within the landing site's covering region at the last time step (t^f) in that arc.

$$(x_{t_f} - \tilde{x})^2 + (y_{t_f} - \tilde{y})^2 + h_{t_f}^2 \leq \tilde{r}^2 \tag{5.38}$$

As discussed in Section 5.4, the glider's EOMs have been linearised around some steady-state conditions. In order to reduce the errors associated to the linearisation we introduce the so-called small perturbation constraints (Inequalities (5.39) and (5.40)) on the control variables C_L (lift coefficient) and μ (roll angle). Preliminary experiments showed that these constraints helped reducing the linearisation errors without overly compromising our algorithm's performance.

$$C_{L,eq} - \delta \leq C_{L,t} \leq C_{L,eq} + \delta, \forall t \in T \tag{5.39}$$

$$\mu_{eq} - \delta \leq \mu_t \leq \mu_{eq} + \delta, \forall t \in T \tag{5.40}$$

From Constraints (5.31–5.40), we are able to define our SOCP TO subproblems as follows. We seek to minimise the error associated with the relaxation of the discretised dynamic represented by Equations (5.30–5.31) in the objective function (5.41). If the end point of arc (i, j) belongs to the set of waypoints, our optimisation subproblem includes visiting Constraints (5.34–5.37). Otherwise, if the ending point is a landing

site, i.e., $j \in L$, our subproblem includes Constraint (5.38) instead. We also include the small perturbartion Constraints (5.39–5.40). Finally, Constraint (5.42) defines the domain of the optimisation variables.

$$
\begin{aligned}
SOCP \quad \min \quad & \varepsilon && (5.41)\\
\text{s.t.} \quad & (5.31\text{–}5.33)\\
& (5.34\text{–}5.37) \textbf{ or } (5.38)\\
& (5.39\text{–}5.40)\\
& \varepsilon \in \mathbb{R}_{\geq 0}, \mathbf{y}_t \in \mathbb{R}^6, \mathbf{u}_t \in \mathbb{R}^2, \forall t \in T. && (5.42)
\end{aligned}
$$

5.4.2 Sequential trajectory optimisation heuristic

This section presents the STO heuristic developed to find feasible trajectories, i.e., feasible states and controls for a glider flying through several waypoints. A feasible trajectory is basically an infinite-dimensional problem defining the state and control of a dynamical system. According to Zhou et al. (2017), even constructing a feasible trajectory under nonlinear constraints is a challenging problem.

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 $r = (i_0, i_1, \ldots, i_l), i_1, \ldots, i_{l-1} \in V$ and $i_l \in L$. The trajectory of a glider flying through route r is then divided into $n_r - 1$ phases 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 explained in Section 5.4.1. The minimum flight time on each phase is computed heuristically. 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.

Figure 5.4 illustrates the procedure described above. In this example, we are asked to find a feasible trajectory for the route $r = (0, 1, 2, l)$, where $1, 2 \in V$ and $l \in L$. The initial conditions on arc $(0, 1)$ are set such that $\mathbf{y}^o_{01} = \mathbf{y}^o$, $\mathbf{u}^o_{01} = \mathbf{u}^o$ and $\tau^o_{01} = 0$, where $\mathbf{y}^o$ and $\mathbf{u}^o$ are known beforehand (Figure 5.4(a)). The shortest flight duration $(\tau^f_{01} - \tau^o_{01})$ on arc $(0, 1)$ is computed heuristically and its corresponding trajectory $(\mathbf{y}_{01}(\tau_{01}), \mathbf{u}_{01}(\tau_{01}))$ is solved to optimality as explained in Section 5.4.1. If no optimal trajectory can be found for arc $(0, 1)$ with a flight duration smaller than a given threshold (FT_{Max}), the algorithm stops and the route is considered infeasible. Otherwise, the solution corresponding to the arc $(0, 1)$ is stored and the algorithm proceeds to the next arc, i.e., $(1, 2)$. The continuity of the trajectory along route r is maintained by setting the initial conditions of arc $(1, 2)$ equal to the final conditions of arc $(0, 1)$ as in Figure 5.4(b). After the last

arc $(2, l)$ is processed (Figure 5.4(c)), the total flight time corresponding to route r can be computed as $\tau^f_{2l} - \tau^o_{01}$.

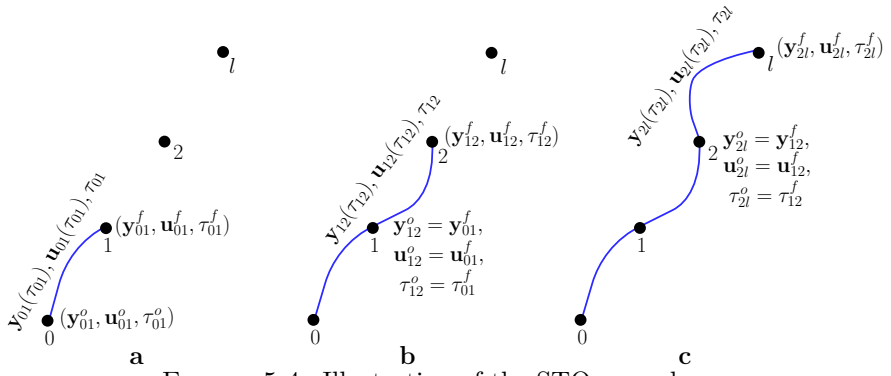


FIGURE 5.4: Illustration of the STO procedure.

Algorithm 1 illustrates the pseudocode of the procedure to find feasible trajectories for a glider. The algorithm starts with the initialisation of the auxiliary variables $\bar{\mathbf{y}}$, $\bar{\mathbf{u}}$ and $\bar{\tau}$, and the initialisation of the route cost $f(r)$ (line 1). The main loop of our heuristic iterates over the arcs of the route r (lines 3–15). The state and control variables associated with arc $(i, j) \in r$ are initialised with an empty value (line 4). Next, the initial conditions ($\mathbf{y}^o_{ij}$ and $\mathbf{u}^o_{ij}$), the initial and final times (τ^o_{ij} and τ^f_{ij}), the error variable (ε_{ij}), the cost (c_{ij}) and the status associated with arc (i, j) are initialised (lines 5–6). The loop on lines (7–9) attempts to optimise the trajectory of the glider associated to route r between waypoints i and j. While an optimal trajectory is not found and the final flight time on arc (i, j) is smaller than a threshold value FT_{Max}, the flight time is incremented by one unit of time (line 8). The TO model is then solved to optimality for a fixed τ^f_{ij}, returning the optimal trajectory between waypoints i and j (represented by $\mathbf{y}_{ij}(\tau_{ij})$ and $\mathbf{u}_{ij}(\tau_{ij})$), the status of the optimisation and the error ε_{ij} associated with the trajectory. If an optimal solution is obtained, the route cost is incremented and the auxiliary coupling variables $\bar{\mathbf{y}}$, $\bar{\mathbf{u}}$ and $\bar{\tau}$ are updated (lines 11–12) thus maintaining the continuity of the trajectory associated with route r. Otherwise, the algorithm terminates and a very large route cost is returned (line 14). If an optimal trajectory is found for every arc of route r, the algorithm returns the real solution cost of r (line 15).

5.5 Proposed matheuristic algorithm

This section describes the proposed matheuristic algorithm, denoted as ILS-STO, for solving the GRTOP defined in Section 5.3. The developed approach consists of a multi-start ILS combined with a Randomized Variable Neighbourhood Descent (RVND) procedure (Subramanian, 2012), a Set Partitioning (SP) integer programming formulation and the STO algorithm described in Section 5.4.

Algorithm 1 `STO`

1: Procedure `STO`(r, FT_{Max})
2: $\bar{\mathbf{y}} \leftarrow \mathbf{y}_o$; $\bar{\mathbf{u}} \leftarrow \mathbf{u}_o$; $\bar{\tau} \leftarrow 0$; $f(r) \leftarrow 0$
3: **for** each arc (i, j) in route r **do**
4: $\quad \mathbf{y}_{ij}(\tau_{ij}) \leftarrow$ `NULL`; $\mathbf{u}_{ij}(\tau_{ij}) \leftarrow$ `NULL`
5: $\quad \mathbf{y}^o_{ij} \leftarrow \bar{\mathbf{y}}$; $\mathbf{u}^o_{ij} \leftarrow \bar{\mathbf{u}}$; $\tau^o_{ij} \leftarrow \bar{\tau}$; $\tau^f_{ij} \leftarrow \bar{\tau}$; $\varepsilon_{ij} \leftarrow 0$; $c_{ij} \leftarrow \infty$
6: $\quad$ *status* $\leftarrow$ not optimal
7: $\quad$ **while** *status* $\neq$ optimal **and** $\tau^f_{ij} < FT_{Max}$ **do**
8: $\quad\quad \tau^f_{ij} \leftarrow \tau^f_{ij} + 1$
9: $\quad\quad [status, \varepsilon_{ij}, \mathbf{y}_{ij}(\tau_{ij}), \mathbf{u}_{ij}(\tau_{ij})] \leftarrow$ `TrajectoryOptimisation`$(\tau^f_{ij}, \varepsilon_{ij})$
10: $\quad$ **if** *status* = optimal **then**
11: $\quad\quad f(r) \leftarrow f(r) + \tau^f_{ij} + \varepsilon_{ij}$
12: $\quad\quad \bar{\mathbf{y}} \leftarrow \mathbf{y}^f_{ij}, \bar{\mathbf{u}} \leftarrow \mathbf{u}^f_{ij}, \bar{\tau} \leftarrow \tau^f_{ij}$
13: $\quad$ **else**
14: $\quad\quad$ **return** ∞
15: **return** $f(r)$
16: **end** `STO`.

For a given arc $(i, j) \in A$, let $\tilde{d}_{ij} = ||(\bar{x}_i, \bar{y}_i) - (\bar{x}_j, \bar{y}_j)||_2$ be an estimate flight distance and $\tilde{v}_{ij}$ be the equilibrium flight velocity (v_{eq}, as computed in Section 5.2.1) between waypoints i and j. We estimate the cost between waypoints i and j as $\tilde{t}_{ij} = \tilde{d}_{ij}/\tilde{v}_{ij}$. Hence, the estimate cost of an arbitrary route r can be defined as in Equation (5.43).

$$\tilde{f}(r) = \sum_{(i,j) \in r} \tilde{t}_{ij} \tag{5.43}$$

Note that determining the value of $\tilde{f}(r)$ is trivial since each $\tilde{t}_{ij}, (i, j) \in A$, can be precomputed in constant time and only once. Let us denote $n_g = |G|$ as the number of available gliders. The estimate cost of a solution s is thus computed as $\tilde{f}(s) = \max_{r=1,\dots n_g}\{\tilde{f}(r)\}$.

Computing the actual cost time between waypoints i and j is computationally expensive, because it requires solving a non-trivial TO problem, as discussed in Section 5.4. Therefore, we resort to the estimation described above while generating an initial solution and performing local search in order to improve the scalability of the method. Yet, we are still forced at a certain point to compute the actual cost $f(r)$ of each route r in a given solution s (i.e., $f(s) = \max_{r=1,\dots n_g}\{f(r)\}$). In particular, we chose to limit this computation to local optimal solutions associated with the estimate costs.

Algorithm 2 shows the pseudocode of ILS-STO. The matheuristic procedure performs I_R restarts (lines 3–17) where at each of them an initial solution is generated using a greedy randomised insertion heuristic (line 4) whose a detailed description can be found in Section 5.5.1. Preliminary experiments showed that by performing a umber of restarts it is possible to improve the overall performance of the algorithm with respect to the obtained objective function values. In our constructive procedure, 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 (line 8) and perturbation (line 14) mechanisms. The first is explained in Section 5.5.2, while the latter is described in Section 5.5.3. If a solution s is improved (with respect to $f(.)$) after the local search phase (which is performed with respect to $\tilde{f}(.)$), then the best current solution is updated (lines 10–12). Next, the pool of routes (*RoutePool*) is updated by adding the feasible routes from s (line 13). The best solution found after each restart is possibly updated in lines 16–17. Finally, the algorithm tries to find the optimal combination of the feasible routes stored in *RoutePool* by solving a SP-based problem (see Section 5.5.4) using a general purpose MILP solver (line 18). Figure 5.5 depicts the procedure described in this section.

Algorithm 2 ILS-STO

```
 1: Procedure ILS-STO(I_R, I_ILS, FT_Max, n_g)
 2: f* ← ∞; Routepool ← NULL
 3: for i:=1,...,I_R do
 4:    s ← GenerateInitialSolution(n_g,seed)
 5:    s' ← artificial solution with very large flight cost  (f(s') ← ∞)
 6:    iter ← 0
 7:    while iter ≤ I_ILS do
 8:       s ← LocalSearch(s)
 9:       f(s) ← max_{r∈s}{f(r) ← STO(r,FT_Max)}
10:       if f(s) < f(s') then
11:          s' ← s
12:          iter ← 0
13:       RoutePool ← RoutePool ∪ {feasible routes from s}
14:       s ← Perturbation(s')
15:       iter ← iter + 1
16:    if f(s') < f* then
17:       s* ← s'; f* ← f(s')
18: s* ← SP(s*, RoutePool)
19: return s*
20: end ILS-STO.
```

5.5.1 Initial solution

The purpose of our initial solution method (`GenerateInitialSolution`) is to find feasible solutions with relatively simple algorithms. We present two different insertion procedures, namely, a Sequential Insertion (SI) method and a Parallel Insertion (PI) method. These are explained in more details in Section 5.5.2. Moreover, we apply two different insertion criteria, namely, a Cheapest Insertion (CI) criterion and a Modified Nearest Insertion (MNI) criterion. These are detailed in Section 5.5.1.1.

Algorithm 3 presents the pseudocode of our constructive procedure. The inputs of our initial solution procedure are the number of available gliders (n_g) and a random seed. First, the maximum number of waypoints in a route is estimated (line 2). A Candidate

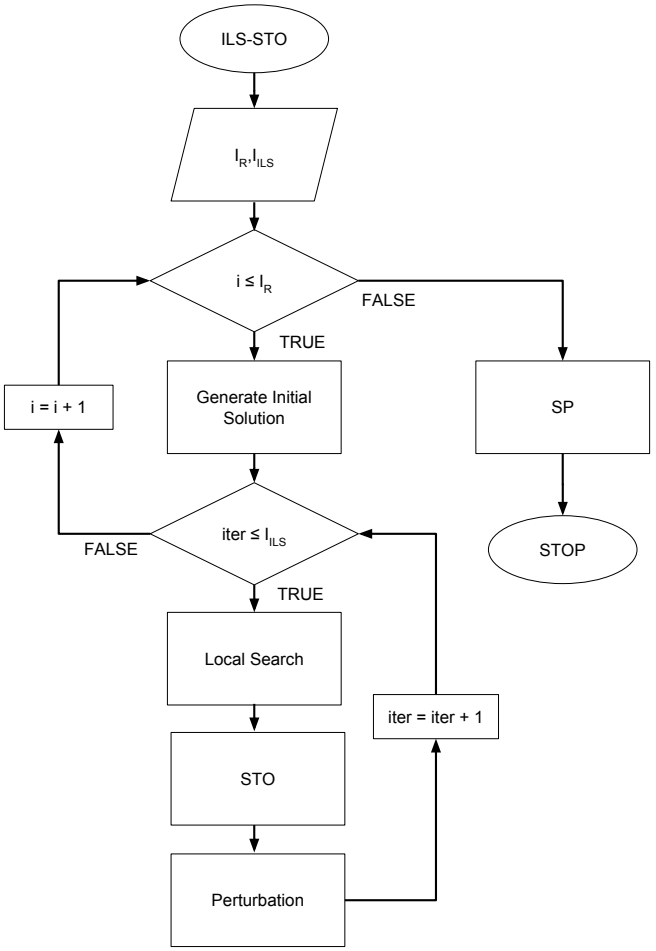


FIGURE 5.5: Flowchart of the ILS-STO algorithm.

List (CL), containing all waypoints from V, and an empty solution s are initialised (lines 3–4). Next, the launching point is added to the beginning of each initial route (lines 5–6). The insertion criterion and strategy are selected at random (lines 7–8). If SI is chosen, the sequential insertion procedure is called with the insertion criterion as a parameter (lines 9–10). Otherwise, the parallel insertion method is called (12). Finally, a randomly chosen landing site is added to the end of each route (line 13–14).

5.5.1.1 Insertion criteria

The cost of inserting an unvisited waypoint $k \in CL$ in a given route r assigned to a glider is computed as in Equation (5.44), according to the selected criterion. If the CI criterion is chosen, the cost of inserting a waypoint k in a route r is computed between every pair of adjacent points i and j belonging to r. If the MNI criterion is chosen, the cost of inserting a waypoint $k \in CL$ after every element i belonging to r is computed. The selected movement is always the one associated with the least-cost insertion, i.e.,

Algorithm 3 `GenerateInitialSolution`

```
1: Procedure GenerateInitialSolution(n_g,seed)
2: r_max ← 2 + ⌈|V|/n_g⌉;
3: Initialize the CL;
4: Let s = {r^1, ..., r^{n_g}} be a solution with n_g empty routes;
5: for r = 1 ... n_g do
6:     s^r ← 0;
7: InsertionCriterion ← CI or NI (chosen at random);
8: InsertionStrategy ← SI or PI (chosen at random);
9: if InsertionStrategy = SI then
10:    s ← SequentialInsertion(s, r_max, CL, InsertionCriterion);
11: else
12:    s ← ParallelInsertion(s, r_max, CL, InsertionCriterion);
13: for r = 1 ... n_g do
14:    s^r ← k ∈ L (chosen at random);
15: return s;
16: end GenerateInitialSolution.
```

$\min\{g(k,r), k \in CL\}$.

$$g(k,r) = \begin{cases} \tilde{t}_{ik} + \tilde{t}_{kj} - \tilde{t}_{ij}, & \text{if InsertionCriterion = CI} \\ \tilde{t}_{ik}, & \text{if InsertionCriterion = MNI} \end{cases} \tag{5.44}$$

5.5.1.2 Insertion strategy

In SI, waypoints are inserted in a single route at each iteration. The pseudocode of SI is shown in Algorithm 4. While the CL is not empty and there is at least one waypoint that can be added to the current partial solution (line 2), each candidate is added to one route according to the selected insertion criterion (lines 3–7). We avoid adding waypoints to routes which correspond to the current makespan (line 4).

Algorithm 4 `SequentialInsertion`

```
1: Procedure SequentialInsertion(s, r_max, CL, InsertionCriterion)
2: while CL ≠ ∅ do
3:     for every route r ∈ s and CL ≠ ∅ do
4:         if |r| ≤ r_max and r ≠ makespan then
5:             k' ← arg min_k {g(k,r) : k ∈ CL};
6:             r ← r ∪ {k'};
7:             Update CL;
8: return s;
9: end SequentialInsertion.
```

In PI, all available gliders and all unvisited waypoints are considered while evaluating the least-cost insertion. Algorithm 5 presents the pseudocode of PI. While the CL is not empty, all insertions are evaluated using the selected insertion criterion and the waypoint associated with the least-cost one is included in the corresponding route (lines 2–5). Routes associated with the current makespan are also avoided in PI.

Algorithm 5 `ParallelInsertion`

1: Procedure `ParallelInsertion`$(s, r_{\max}, CL, InsertionCriterion)$
2: **while** $CL \neq \emptyset$ **do**
3: $k', r' \leftarrow \arg\min_{k,r} \{g(k,r) : k \in CL, r \in s : |r| \leq r_{max}\}$;
4: $r' \leftarrow r' \cup \{k'\}$;
5: Update CL;
6: **return** s;
7: **end** `ParallelInsertion`.

5.5.2 Local search

We apply a RVND procedure, based on the method presented by Subramanian (2012), for performing the local search. Let $\mathcal{N}$ be the set of inter-route neighbourhoods, that is, those involving more than one route. RVND starts by randomly choosing a neighbourhood $\eta \in \mathcal{N}$, and determining the best improving move. If no improvements have been found, a neighbourhood other than η is selected at random until all neighbourhoods fail to improve the best current solution. In case of improvement, RVND restarts after performing a search in the modified routes also using a RVND scheme, but with intra-route neighbourhoods, that is, those involving moves within the route.

Algorithm 6 shows the `LocalSearch` method. The algorithm starts by initialising the neighbourhood list $\mathcal{N}$. The main loop of the algorithm runs until $\mathcal{N}$ is empty (lines 3–19). The loop starts by choosing a inter-route operator $\eta \in \mathcal{N}$ at random (line 4). Next, the best move is selected (line 5) and the least-cost solution is updated accordingly (lines 6–7). Let $\mathcal{N}'$ be an intra-route neighbourhood list. We perform an intensification step while $\mathcal{N}'$ is not empty (lines 9–16). Within this loop, a neighbourhood $\eta' \in \mathcal{N}'$ is randomly selected and a local search is performed until no more improvements to the best current solution are found. Both inter- and intra-route neighbourhood sets are reinitialised every time an improvement is made (lines 14 and 17, respectively).

5.5.2.1 Inter-route operators

The inter-route search consists of three basic operators, namely, `Shift(`k`)`, `Swap(`k_1, k_2`)` and `Cross`. These neighbourhoods are only applied when at least one of the routes has an estimate cost that is equal to the estimate maximum route cost of the current solution. For example, in `Shift(`k`)`, we only consider moving k adjacent waypoints from the route whose estimate cost matches the maximum but not the other way around as there cannot be any improvements in the solution cost by inserting a waypoint in the route associated with the maximum cost. In what follows, we provide some details on the implementation of the inter-route operators.

Algorithm 6 `LocalSearch`

```
1: Procedure LocalSearch(s)
2: Initialize the inter-route neighbourhood set (𝒩);
3: while 𝒩 ≠ ∅ do
4:    Choose a neighbourhood η ∈ 𝒩 at random;
5:    Find the best neighbour s′ of s ∈ η;
6:    if f(s′) < f(s) then
7:       s ← s′;
8:       Initialize the intra-route neighbourhood set (𝒩′);
9:       while 𝒩′ ≠ ∅ do
10:         Choose a neighbourhood η′ ∈ 𝒩′ at random;
11:         Find the best neighbour s′ of s ∈ η′;
12:         if f(s′) < f(s) then
13:            s ← s′;
14:            Update 𝒩′; {𝒩′ is populated with all inter-route neighbourhood structures}
15:         else
16:            Remove η′ from 𝒩′;
17:      Update 𝒩; {𝒩 is populated with all inter-route neighbourhood structures}
18:   else
19:      Remove η from 𝒩;
20: return s;
21: end LocalSearch.
```

- `Shift(`k`)`: This operators consists of moving k adjacent waypoints from a route r_1 to a route r_2. Algorithm 7 illustrates the pseudocode of `Shift(`k`)`. We recall that moves are not evaluated if r_2 corresponds to the current makespan.

Algorithm 7 `Shift(`k`)`

```
1: Procedure Shift(k, s)
2: for r1 = 1 ... n_g do
3:    for r2 = 1 ... n_g do
4:       if r1 ≠ r2 then
5:          for every waypoint i ∈ r1 do
6:             for each position j ∈ r2 do
7:                Evaluate cost f̃(s′) of transferring k adjacent waypoints ∈ r1 to position j in r2;
8:                if f̃(s′) < f̃* then
9:                   f̃* ← f̃(s′);
10:                  s* ← s′;
11: if f̃* < f̃(s) then
12:    s ← s*;
13: return s;
14: end Shift(k).
```

- `Swap(`k_1, k_2`)`: This operators consists of exchanging k_1 adjacent waypoints from a route r_1 with k_2 adjacent waypoints from a route r_2. This move only occurs if at least r_1 or r_2 correspond to the current makespan. Algorithm 8 illustrates the pseudocode of `Swap(`k_1, k_2`)`.

- `Cross`: This operator consists of removing the arc between adjacent waypoints k and l, belonging to a route r_1, and the one between k' and l', from a route r_2. Next, an arc is inserted connecting k and l' and another is inserted linking k' and l. Moves are evaluated only if one of the routes correspond to the makespan. Algorithm 9 illustrates the pseudocode of `Cross`.

Algorithm 8 `Swap`(k_1, k_2)

```
1: Procedure Swap(k1, k2, s)
2: for r1 = 1 ... ng do
3:    for r2 = r1 + 1 ... ng do
4:       for every k1 adjacent waypoints ∈ r1 do
5:          for every k2 adjacent waypoints ∈ r2 do
6:             Evaluate cost f̃(s') of exchanging k1 waypoints ∈ r1 with k2 waypoints ∈ r2;
7:             if f̃(s') < f̃* then
8:                f̃* ← f̃(s');
9:                s* ← s';
10: if f̃* < f̃(s) then
11:    s ← s*;
12: return s;
13: end Swap(k1, k2).
```

Algorithm 9 `Cross`

```
1: Procedure Cross(s)
2: for r1 = 1 ... ng do
3:    for r2 = 1 ... ng do
4:       if r1 ≠ r2 then
5:          for every position i in r1 do
6:             for every position j in r2 do
7:                Evaluate the cost f̃(s') of removing arc (k, l) ∈ r1, associated with positions i and i+1,
                  and arc (k', l') ∈ r2, associated with positions j − 1 and j, from the current solution,
                  and inserting arcs (k, l') and (k', l);
8:                if f̃(s') < f̃* then
9:                   f̃* ← f̃(s');
10:                  s* ← s';
11: if f̃* < f̃(s) then
12:    s ← s*;
13: return s;
14: end Cross.
```

5.5.2.2 Intra-route operators

Solutions associated with our intra-route operators are evaluated as in the TSP. After performing the intra-route local search, the algorithm updates, if necessary, the maximum route cost. Our intra neighbourhood structures are defined as follows:

- `Or-opt(`k`)`: This operator consists of removing and reinserting k adjacent waypoints in a different position of the current route. Algorithm 10 illustrates the pseudocode of `Or-opt(`k`)`,
- `2-opt`: This operator consists of deleting two non-adjacent arcs and adding another two in such a way that a new route is generated. Algorithm 11 illustrates the pseudocode of `2-opt`,
- `Exchange`: This operator consist of exchanging the positions of two waypoints from the same route. Algorithm 12 illustrates the pseudocode of `Exchange`.

Algorithm 10 `Or-opt(`k`)`

```
1: Procedure Or-opt(k, s)
2: for r = 1 ... n_g do
3:    if r is the current makespan then
4:       for every position i in r do
5:          for every position j in r do
6:             if i ≠ j and k adjacent waypoints can be moved from i to j then
7:                Evaluate cost f̃(s') of reallocating k adjacent waypoints from position i to position j;
8:                if f̃(s') < f̃* then
9:                   f̃* ← f̃(s');
10:                  s* ← s';
11:      if f̃* < f̃(s) then
12:         s ← s*;
13:         r ← 1;
14: return s;
15: end Or-opt(k).
```

Algorithm 11 `2-opt`

```
1: Procedure 2-opt(s)
2: for r = 1, ..., n_g do
3:    if r is the current makespan then
4:       for i = 1 ... |r| − 3 do
5:          for j = i + 3...|r| do
6:             Evaluate cost f̃(s') of deleting arcs (k, l), relative to positions i and i + 1, and (k', l'),
               relative to positions j − 1 and j, then adding arcs (k, k') and (l, l'), and reversing the arcs
               between l and k';
7:             if f̃(s') < f̃* then
8:                f̃* ← f̃(s');
9:                s* ← s';
10:      if f̃* < f̃(s) then
11:         s ← s*;
12:         r ← 1;
13: return s;
14: end 2-opt.
```

Algorithm 12 `Exchange`

```
1: Procedure Exchange(s)
2: for r = 1, ..., n_g do
3:    if r is the current makespan then
4:       for i = 2 ... |r| − 2 do
5:          for j = i + 1...|r| − 1 do
6:             Evaluate cost f̃(s') of swapping waypoints k and l relative to positions i and j, respec-
               tively;
7:             if f̃(s') < f̃* then
8:                f̃* ← f̃(s');
9:                s* ← s';
10:      if f̃* < f̃(s) then
11:         s ← s*;
12:         r ← 1;
13: return s;
14: end Exchange.
```

5.5.3 Perturbation mechanism

Two perturbation mechanisms were adopted. Whenever the `Perturbation` function is called, one of the moves below is randomly selected. The only special case happens if a

solution s consists of a single route. In this case, part of this route is randomly sorted so that a new route is generated.

- `M_Swap(1,1)`: Consists of performing multiple random `Swap(1,1)` moves.
- `M_Shift(1,1)`: Consists of performing multiple random `Shift(1,1)` moves.

5.5.4 A set partitioning-based approach

Let $\mathcal{R}$ be the set of all feasible routes with associated cost $c_j, j \in \mathcal{R}$. Also, let $\mathcal{R}_i \subseteq \mathcal{R}$ be the set of all feasible routes containing the waypoint $i \in V$. Define y_j as a binary variable that assumes value 1 if a route $j \in \mathcal{R}$ is in the solution and 0 otherwise, and C_{max} as the variable associated with the maximum route cost. One can write a SP-based formulation for the GRTOP described in Section 4.3 as follows:

$$\min \quad C_{max} \tag{5.45}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{R}_i} y_j = 1 \qquad i \in V \tag{5.46}$$

$$\sum_{j \in \mathcal{R}} y_j \leq n_g \tag{5.47}$$

$$C_{max} \geq c_j y_j \qquad j \in \mathcal{R} \tag{5.48}$$

$$y_j \in \{0,1\} \qquad j \in \mathcal{R} \tag{5.49}$$

The objective function (5.45) minimises the maximum route cost which is bounded by Constraints (5.48). Constraints (5.46) state that there should be exactly one route associated with each waypoint $i \in V$. Constraint (5.47) imposes an upper bound on the number of gliders. Constraints (5.49) define the domain of the variables.

Since it is prohibitively expensive to determine all feasible routes, we actually solve a restricted version of model (5.45–5.49) which is composed of a subset of routes $\mathcal{R}' \subseteq \mathcal{R}$ obtained by ILS-STO.

5.6 Computational experiments

ILS-STO was coded in C++ and 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). The software CPLEX 12.7 was used, in its standard configuration, to solve the SOCP subproblems described in Section 5.4.1.

In this section, the method for generating instances closely follows the procedure presented in Section 4.5.1. We have generated instances having $n \in \{10, \ldots, 50\}$ waypoints

and $m \in \{3, 4, 5\}$ landing zones. The number of gliders is computed by $n_g = \lfloor n/2 \rfloor, n_g \leq 10$. Table 5.2 shows the values of the parameters defining waypoints and landing zones as well as the dimensions of the area containing them (units are indicated as necessary). 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 $10km^2$. Applying nomenclature similar to the one used in Section 4.5.1, we will refer to the instances created in this section as *large range* instances (represented by "L" in the instance name). During all of our experiments, the discretisation size N has been set to 50.

TABLE 5.2: Parameters defining the geometry of waypoints and landing zones.

Parameter	a	b	Parameter	a	b
$\bar{x}(km)$	0	10	$\tilde{x}(km)$	0	10
$\bar{y}(km)$	0	10	$\tilde{y}(km)$	0	10
$\bar{h}$	0	0	$\tilde{h}$	0	0
$\bar{r}(m)$	10	25	$\tilde{r}(m)$	10	25
$\underline{h}(m)$	50	100	$x_o(km)$	0	10
$\bar{\bar{h}}(m)$	200	300	$y_o(km)$	0	10

We ran ILS-STO ten times for each instance and the results obtained are reported in Table 5.3. 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 5.3: 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
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

continues in the next page

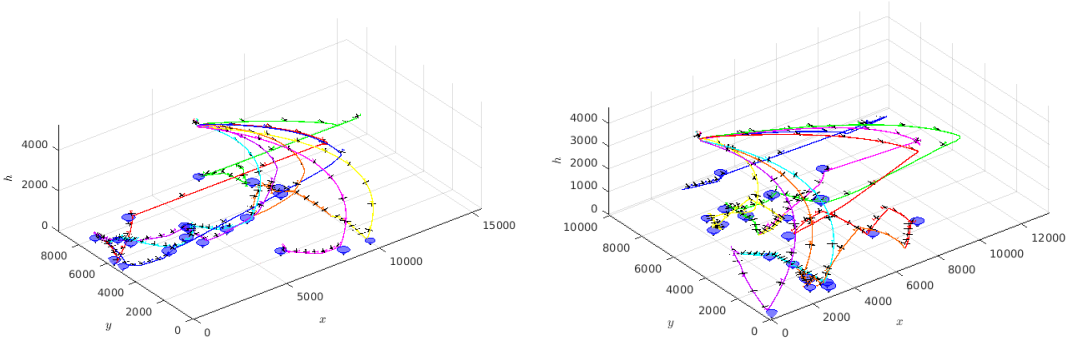
Table 5.3 – continued from previous page

Instance	Best Obj.	Avg. Obj.	Gap(%)	Time(s)	Error	CPU(s)	TO(%s)
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 5.3, 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 each solution takes a value of $\varepsilon = 51.17$, on average. In Section 4.5, 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 `grtopL_104_1` and `grtopL_104_4`, the average gap was larger than 12%. Figure 5.6 shows feasible solutions of two different instances.



(A) Solution of `grtopL_205_1`. (B) Solution of `grtopL_205_5`.

FIGURE 5.6: Depiction of the optimal solutions of two large range instances.

Table 5.4 shows the effect of applying the SP-based approach after the multi-start ILS procedure. In this table, the average objective function values before and after solving the SP formulation are reported. Column **Avg. Obj.** shows the average best found solution cost before solving the SP formulation, while column **SP Obj.** reports the optimal objective function value produced by solving the SP problem. Column **Improved** shows how much the local optimal cost is improved, on average, by using the SP module. This improvement is calculated as $100 \times [(\text{Avg. Obj.} - \text{Avg. SP})/\text{Avg. Obj.}]$. Finally, column **Success** reports the average success rate of the SP step on improving the solutions found by ILS-STO.

Overall, the SP-based approach improved the objective function value by 0.77% on approximately 40% of the instances, on average. However, if we only consider the larger instances (with 50 waypoints), the improvement on the optimal value increases to 1.31% on 64% of the cases, on average. We highlight that the time required to solve the SP formulation is rather negligible when compared against the overall running times of ILS-STO. These results suggest that the SP module is useful in our framework, as it helps the method to improve the average objective function value with little added computational effort, especially on larger instances.

TABLE 5.4: Effect of applying the SP-based approach.

Instance	**Avg. Obj.**	**Avg. SP**	**Improved**	**Success**
grtopL_103_1	674.32	674.32	0.00%	0%
grtopL_103_2	373.51	373.51	0.00%	0%
grtopL_103_3	538.21	538.21	0.00%	0%
grtopL_103_4	399.00	398.73	0.07%	10%
grtopL_103_5	524.89	524.90	0.00%	0%
grtopL_104_1	414.52	414.52	0.00%	0%
grtopL_104_2	418.81	417.42	0.33%	10%
grtopL_104_3	481.39	481.28	0.02%	10%
grtopL_104_4	443.45	442.78	0.15%	10%
grtopL_104_5	422.27	422.27	0.00%	0%
grtopL_105_1	469.63	469.63	0.00%	0%
grtopL_105_2	373.46	367.12	1.70%	30%
grtopL_105_3	346.34	346.34	0.00%	0%
grtopL_105_4	389.14	382.48	1.71%	70%
grtopL_105_5	416.75	409.29	1.79%	90%
grtopL_203_1	349.92	344.66	1.50%	60%
grtopL_203_2	453.83	453.83	0.00%	0%
grtopL_203_3	413.03	413.03	0.00%	0%
grtopL_203_4	456.62	456.61	0.00%	10%
grtopL_203_5	454.26	454.26	0.00%	0%
grtopL_204_1	412.64	407.71	1.20%	60%
grtopL_204_2	475.13	473.21	0.40%	30%
grtopL_204_3	543.11	543.11	0.00%	0%
grtopL_204_4	480.46	480.17	0.06%	20%
grtopL_204_5	362.92	356.46	1.78%	90%
grtopL_205_1	361.31	359.96	0.37%	10%
grtopL_205_2	429.45	429.24	0.05%	10%
grtopL_205_3	440.20	440.20	0.00%	0%
grtopL_205_4	396.84	396.84	0.00%	0%
grtopL_205_5	396.63	396.63	0.00%	0%
grtopL_303_1	596.05	593.73	0.39%	30%
grtopL_303_2	507.25	507.25	0.00%	0%
grtopL_303_3	393.47	389.54	1.00%	60%
grtopL_303_4	436.05	433.25	0.64%	50%
grtopL_303_5	438.65	436.62	0.46%	10%
grtopL_304_1	406.88	401.60	1.30%	100%
grtopL_304_2	381.04	376.80	1.11%	100%
grtopL_304_3	377.88	374.75	0.83%	20%
grtopL_304_4	404.03	390.00	3.47%	90%
grtopL_304_5	422.31	422.31	0.00%	0%
grtopL_305_1	504.22	504.22	0.00%	0%
grtopL_305_2	416.08	411.22	1.17%	90%
grtopL_305_3	444.09	444.09	0.00%	0%
grtopL_305_4	366.87	363.91	0.81%	80%

continues in the next page

Table 5.4 – continued from previous page

Instance	Avg. Obj.	Avg. SP	Improved	Success
grtopL_305_5	411.80	409.32	0.60%	30%
grtopL_403_1	449.17	443.15	1.34%	90%
grtopL_403_2	483.69	483.66	0.01%	10%
grtopL_403_3	409.65	409.65	0.00%	0%
grtopL_403_4	487.56	477.75	2.01%	100%
grtopL_403_5	408.51	398.00	2.57%	100%
grtopL_404_1	443.86	443.67	0.04%	40%
grtopL_404_2	474.87	474.87	0.00%	0%
grtopL_404_3	388.37	384.43	1.01%	70%
grtopL_404_4	488.80	481.29	1.54%	100%
grtopL_404_5	523.30	517.51	1.11%	60%
grtopL_405_1	526.68	526.68	0.00%	0%
grtopL_405_2	397.44	392.64	1.21%	100%
grtopL_405_3	427.46	425.23	0.52%	70%
grtopL_405_4	483.55	468.53	3.11%	100%
grtopL_405_5	456.31	452.37	0.86%	70%
grtopL_503_1	425.57	419.18	1.50%	90%
grtopL_503_2	393.29	387.03	1.59%	90%
grtopL_503_3	449.18	447.90	0.29%	20%
grtopL_503_4	494.21	494.05	0.03%	30%
grtopL_503_5	431.03	418.76	2.85%	90%
grtopL_504_1	400.93	399.22	0.43%	20%
grtopL_504_2	448.69	438.66	2.23%	90%
grtopL_504_3	473.04	473.04	0.00%	0%
grtopL_504_4	428.03	418.18	2.30%	100%
grtopL_504_5	460.33	456.02	0.93%	80%
grtopL_505_1	388.35	384.66	0.95%	40%
grtopL_505_2	434.93	429.79	1.18%	90%
grtopL_505_3	455.51	451.91	0.79%	80%
grtopL_505_4	380.03	364.30	4.14%	90%
grtopL_505_5	418.05	416.08	0.47%	50%

Figure 5.7 depicts the effect of varying the maximum allowed number of gliders on the average makespan value for a subgroup of instances. In this figure, the vertical axis shows the value of the makespan, while the horizontal axis shows each instance in the considered subgroup. Each category represents the maximum allowed number of gliders in the fleet (i.e., $1, 4, 7, 10$ and 13 gliders). We can observe that the average makespan tends to decrease (with exceptions) in many cases with the addition of gliders to the fleet. However, this trend cannot be generalised. For example, if we allow at most 13 gliders in the fleet, the average makespan tends to be larger, for many instances in this subgroup, than if only 10 gliders are allowed. These results suggest that a reasonable fleet size lies between 7 and 10 gliders.

Figure 5.8 shows the effect of varying the maximum allowed number of gliders on the best makespan value for the same subgroup of instances. The results in this figure confirm our analysis based on the average makespan.

Figure 5.9 shows the utilisation of the fleet for different maximum allowed fleet sizes. In the vertical axis we report the values of utilisation, defined as the number of gliders used in the local optimal solution divided by the maximum fleet size. These results indicate

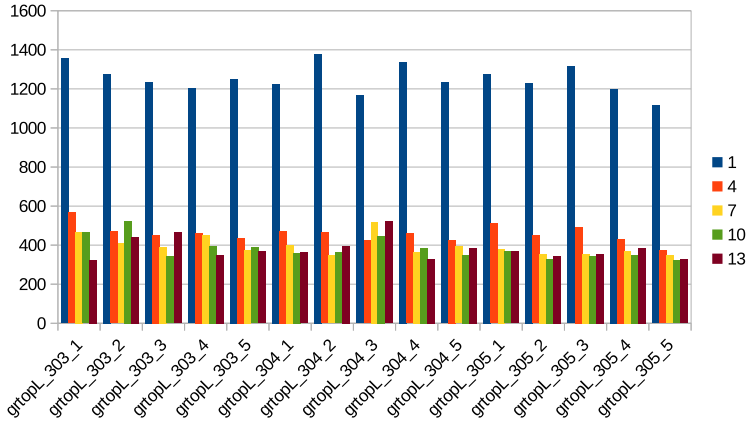


FIGURE 5.7: Average makespan for different maximum allowed number of gliders in a fleet.

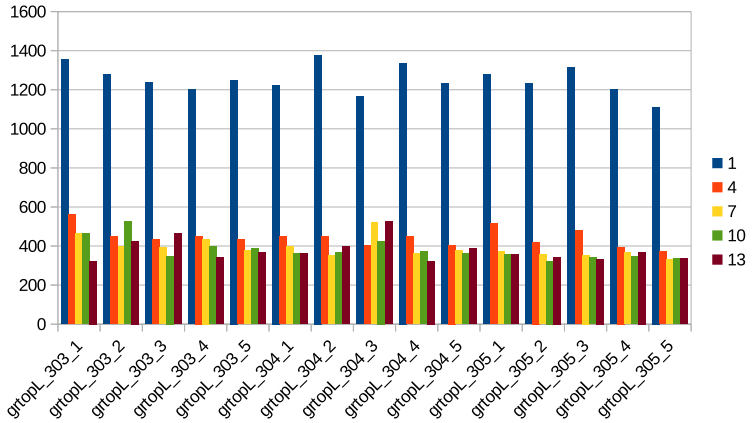


FIGURE 5.8: Best makespan for different maximum allowed number of gliders in a fleet.

that, for most instances in this subgroup, allowing more that 10 gliders in the fleet does not strongly affect the actual number of used gliders.

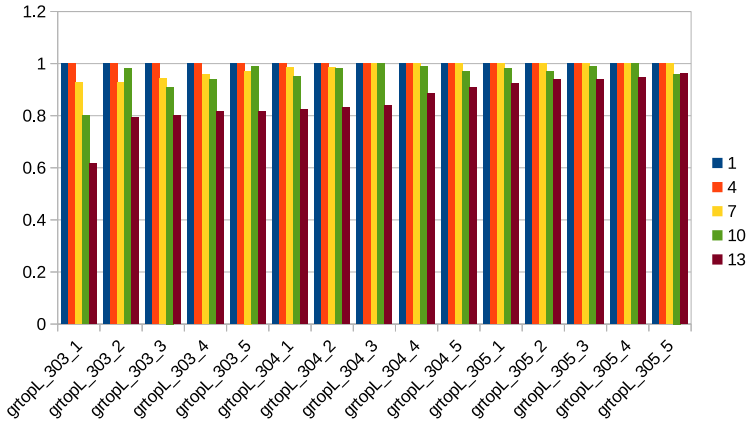


FIGURE 5.9: Utilisation of the fleet for different maximum fleet sizes.

5.7 Conclusions of the chapter

In this chapter, we presented a heuristic algorithm for the GRTOP. We adopted two modifications to this problem's definition. The first modification consists of modelling TO as a multi-phase optimal control problem, where the dynamics is allowed to change within a given route. The second one is concerned with the objective function, in which we seek to minimise the maximum route duration (a.k.a. makespan). 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.

We developed an arc-based Multi-phase Mixed-Integer Trajectory Optimisation Formulation for the GRTOP. In this formulation, each phase of the TO is associated to an arc in an directed graph. Moreover, each arc contains its own state variables, control variables and flight dynamics. However, we did not attempt to implement this formulation, since it is highly nonlinear and nonconvex. Instead, we proposed a heuristic method capable of finding feasible solutions with a small computational effort. 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.

Chapter 6

Conclusions and Future Work

6.1 Research outcomes

In this thesis, we were concerned with problems involving routing and trajectory optimisation of UAVs. This type of problem is more challenging than the classic VRP due the combination of routing, a combinatorial optimisation problem, and optimal control, which is usually nonlinear and nonconvex. Moreover, we focused on exact and heuristic methods for the problem of simultaneously optimising the routes and trajectories of a fleet of aerial gliders.

Our literature review revealed that a formal definition of the UAV Routing and Trajectory Optimisation Problem (UAVRTOP) was yet to be proposed. Most of the research articles that attempted to solve UAV routing and trajectory optimisation problems relied on simplifying assumptions that did not represent its complexity in real world applications. We believe that integrating integrating routing with the flight dynamics of UAVs is necessary to ensure the feasibility of routes in several scenarios. Generally speaking, three main issues arise when dealing with UAVRTOPs: (i) how to tackle the set of the ODEs modelling flight dynamics; (ii) hot to integrate routing and trajectory optimisation in the same framework; and (iii), how to develop a scalable method capable of solving instances in reasonable computing times.

The main contribution of this thesis can be summarized as follows:

- Motivated by a disaster assessment application, we proposed a variant of the UAVRTOP, the so-called Glider Routing and Trajectory Optimisation Problem (GRTOP). In the GRTOP, camera-equipped gliders are due to surveying a number of risky locations in a post-disaster situation. We employed a set of ODEs to represent the motion and flight dynamics of gliders, they are referred to as gliders' EOMs. We introduced a single-phase MINLP for solving the GRTOP. In order

to avoid a nonconvex and nonlinear formulation, we linearised the EOMs around some steady-state conditions. Several integration methods were presented in order to discretise the resulting linear EOMs, namely, an Euler's method, a Trapezoidal method, two Runge-Kutta methods and two Adams-Bashforth methods. In addition, constraints related to the gliders' EOMs were relaxed and a penalty term was added to the objective function, resulting in a more tractable MISOCP formulation. We tested the proposed formulation against a large number of randomly generated instances and real-life based instances. We compared the performance of the model in several scenarios, considering different solvers, different discretisations and an alternative relaxation of the EOMs. To the best of our knowledge, this is one of the first approaches in the literature to integrate routing and trajectory optimisation of UAVs into a single framework.

- Moreover, we developed matheuristic algorithm for solving the GRTOP. The proposed algorithm is based on two main building blocks, namely, a STO heuristic and a ILS-SP matheuristic. We introduced a multi-phase MINLP for the GRTOP. Two main modifications were performed to the original formulation. The first one is the adoption of a multi-phase TO formulation, instead of a single-phase one. This allows for changes in the glider's dynamics within routes. The second one consists of minimising the duration of the longest route (a.k.a. makespan). The so-called STO heuristic was designed for solving multi-phase glider TO problems. STO uses a decomposition scheme, where trajectories are divided into segments according to the arcs of a given route. Next, each arc is solved as a single-phase SOCP trajectory optimisation subproblem. A feasible solution is then constructed by joining together the solutions of each arc in the appropriate order. The second building block of our method is an ILS-SP matheuristic capable of generating sequences that can be evaluated by STO. ILS-SP is mostly based on existing approaches for VRPs.

Both algorithms presented in this thesis present advantages and limitations. The exact method introduced in Chapter 4 is one of the first to tackle this type of problem. Computational experiments with the single-phase model showed that it can be used for solving small sized problems. Even though this formulation is capable of finding optimal solutions, accuracy is limited by small discretisation sizes. In addition, high discretisation sizes are usually associated with long CPU times. On the other hand, our heuristic algorithm is capable of finding feasible solutions with a small computational effort. However, only suboptimal trajectories can be found. Another limitation of our method is related to the linearisation of the glider's EOMs.

Embedding the glider's EOMs directly into an optimisation problem generally leads to non-convex formulations. This work presented a linearisation of the EOM using a first-order Taylor's expansion around steady-state conditions. This linearisation allows to

formulate the resulting TOP as a SOCP and therefore commercial optimisation software such as CPLEX and GUROBI can be used. However, the generated trajectories are highly dependent on steady-state conditions, which means that classical control algorithms might face some difficulties for tracking these trajectories in real life situations.

6.2 Future research

We believe that the work presented here has contributed to the existing literature. So far, our method relies on the linearisation of the glider's EOMs. This research can be extended by considering the original glider's EOMs. The methods presented in this thesis can be used to generate initial guesses fora a MINLP considering the original glider's dynamics.

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.

Regarding the ILS-STO algorithm, our next research steps would include embedding STO into the local search phase of ILS. For example, let $r_1 = (0, 1, 2, 3, 4, 5, 6, 7)$ be the current best solution with cost c_1. Let us suppose that the neighbour solution $r_2 = (0, 1, 2, 3, 4, 6, 5, 7)$ needs to be evaluated, i.e., the solution cost c_2 must be computed. Note that these solutions are identical until the fifth position, therefore STO would run only from the sixth position onwards, since the cost of the segment $0-1-2-3-4$ would remain the same. The challenge of performing such modification to our method consists of integrating the local search and STO without overly compromising the running times of the algorithm.

Both methods presented in this thesis can be easily adapted to other types of vehicles with motion constraints, such as unmanned underwater vehicles and powered UAVs. By replacing the gliders' dynamics with the appropriate dynamics one could easily apply the methodology presented in this thesis for solving many classes of UAVRTOPs.

Appendix A

Appendix

A.1 Methods and applications for the selected literature

Table A.1 highlights the methods and practical applications addressed in the 70 selected papers.

Table A.1: Summary of methods and applications on 70 selected papers.

Authors	Approach	Application
Al-Sabban et al. (2012)	Markov decision process	Path planning in uncertain wind conditions
Babel (2011)	Shortest path algorithms	UAV path planning with obstacles
Babel (2012)	Shortest path algorithms	Path planning in a risk environment
Bae et al. (2015)	Dynamic programming and heuristics	Risk-constrained shortest path for UCAV
Baiocchi (2014)	Heuristic algorithms	Path planning for aerial photography
Bandeira et al. (2015)	Heuristic algorithms	UAV routing for aerial photography
Bednowitz et al. (2012)	Simulation model	UAV routing in dynamic environment
Besada-Portas et al. (2010)	Evolutionary algorithms	Real-time UAV path planning
Besada-Portas et al. (2013)	Evolutionary algorithms	Real-time UAV path planning
Casbeer and Holsapple (2011)	Column generation	UAV TA with precedence
Chakrabarty and Langelaan (2011)	Energy map method	Path planning for soaring UAVs
Chen et al. (2016)	Genetic algorithm	Multi UAV trajectory optimisation
Choe et al. (2016)	Pythagorean hodograph bézier curves	Cooperative path planning
Cobano et al. (2013)	Rapid exploring random trees	Cooperative trajectory optimisation
Cons et al. (2014)	Heuristic algorithms	Integrated TA and path planning
Crispin (2016)	Rapid exploring random trees	Path planning for aerial gliders
Dilão and Fonseca (2013)	Heuristic algorithms	Path planning for a hypersonic glider
Edison and Shima (2011)	Genetic algorithm	Integrated TA and path planning
Enright et al. (2015)	Queueing theory	UAV routing in stochastic environments
Evers et al. (2014)	ILS metaheuristic	UAV orienteering problem with time windows
Faied et al. (2010)	Mixed-Integer Linear Programming	Multi UAV routing problem
Filippis et al. (2011)	Shortest path algorithms	UAV path planning with obstacles
Forsmo (2012)	Mixed-Integer Linear Programming	UAV routing and trajectory optimisation
Fügenschuh and Müllenstedt (2015)	Mixed-Integer Linear Programming	UAV routing and trajectory optimisation
Furini et al. (2016)	Mixed-Integer Linear Programming	Time dependent UAV routing problem
Gottlieb and Shima (2015)	Enumerative and heuristic algorithms	Integrated TA and path planning
Guerriero et al. (2014)	Multi-objective optimisation	UAV routing with time windows
Han et al. (2014)	Dynamic programming	UAV path planning
Henchey et al. (2016)	Enumerative and heuristic algorithms	UAV routing problems
Huang et al. (2016)	Ant colony optimisation	Multi UAV path planning
Hu et al. (2015b)	Ant colony optimisation	UAV task assignment
Jaishankar and Pralhad (2011)	Multi criteria decision analysis	UAV path planning
Jiang and Ng (2011)	Mixed-Integer Linear Programming	Multi UAV routing problem
Kagabo (2010)	Fuzzy Logic	Path planning for aerial gliders
Kivelevitch et al. (2016)	Market-based algorithm	UAVs TA problem
Kumar and Padhi (2013)	Model predictive static programming	UAV trajectory optimisation

Continued on next page

Table A.1 – continued from previous page

Authors	Approach	Application
Kwak et al. (2013)	Heuristic algorithms	Generalised UAVs TA
Levy et al. (2014)	Heuristic algorithms	UAVs routing with refuelling depots
Liu et al. (2013)	Heuristic algorithms	Real-time UAV path planning
Liu et al. (2016)	Collocation interval analysis method	UAV path planning
Manyam et al. (2015)	Lagrangian Relaxation	Multi depot UAVs routing
Mersheeva (2015)	Heuristics and constraint programming	UAV routing in disaster assessment
Mufalli et al. (2012)	Mixed-Integer Linear Programming and heuristics	UAVs routing problems
Murray and Karwan (2010)	Mixed-Integer Linear Programming	UAVs dynamic TA and routing
Murray and Karwan (2013)	Branch-and-bound	UAVs dynamic routing
Myers et al. (2016)	Shortest path algorithm	Real-time UAV path planning
Nguyen et al. (2015)	Look-up tables	Path planning for aerial gliders
Niccolini et al. (2010)	Descriptor functions methodology	Multi UAV TA problem
Park et al. (2012)	Heuristic algorithms	UAV routing
Pepy and Hérissé (2014)	Indirect shooting method	Trajectory optimisation for an aerial glider
Pharpatara et al. (2015)	Geometric path planning	Path planning for a hypersonic glider
Rogowski and Maroński (2011)	Direct pseudospectral method	Trajectory optimisation for an aerial glider
Shanmugavel (2013)	Bayesian rule-based algorithm	UAV path planning
Silva et al. (2015)	Non-linear programming	Trajectory optimisation for an aerial glider
Song et al. (2016)	Mixed-Integer Linear Programming and heuristics	Multi UAV TA problem
Stump and Michael (2011)	Mixed-Integer Linear Programming	Multi UAV routing problem
Sundar and Rathinam (2014)	Heuristic and approximation algorithms	UAV routing with refueling depots
Techy et al. (2010)	Heuristic algorithm	UAV path planning in uniform wind
Thi et al. (2012)	Exact and heuristic algorithms	UAVs task assignment
Vilar and Shin (2013)	Heuristic algorithm	Communication-aware TA problem
Wang et al. (2015)	Heuristic algorithms	Multi UAV TA problem
Wang et al. (2016)	Population-based algorithms	UAV path planning
Wu et al. (2011)	Genetic Algorithm	UAV path planning
Xu et al. (2017)	Gradient-descent algorithm	UAV path planning
Yakıcı (2016)	Ant Colony Optimisation	UAV location and routing problem
Yang et al. (2015)	Heuristic algorithms	UAV path planning
Yomchinda et al. (2016)	Parametrization techniques and heuristics	Aircraft path planning
Zhang et al. (2011)	Differential Evolution Algorithm	UAVs real-time path planning
Zhang et al. (2012)	Heuristic algorithm	UAV routing and trajectory optimisation
Zhang et al. (2014)	Memetic Algorithm	UAV routing problem

Bibliography

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

Al-Sabban, W. H., Gonzalez, L. F., Smith, R. N., and Wyeth, G. F. (2012). Wind-energy based path planning for electric unmanned aerial vehicles using Markov Decision Processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6, Algarve. IEEE.

Alidaee, B., Gao, H., and Wang, H. (2010). A note on task assignment of several problems. *Computers & Industrial Engineering*, 59(4):1015–1018.

Alidaee, B., Wang, H., and Landram, F. (2011). On the Flexible Demand Assignment Problems: Case of Unmanned Aerial Vehicles. *IEEE Transactions on Automation Science and Engineering*, 8(4):865–868.

Alighanbari, M. and How, J. P. (2008). A robust approach to the UAV task assignment problem. *International Journal of Robust and Nonlinear Control*, 18(2):118–134.

AMPL (1998). New in ampl: Statuses. `http://www.ampl.com/NEW/statuses.html`. Accessed: 25/05/2017.

Andelmin, J. and Bartolini, E. (2017). An Exact Algorithm for the Green Vehicle Routing Problem. *Transportation Science*, 51(4):1288–1303.

Ariyur, K. B. and Fregene, K. O. (2008). Autonomous tracking of a ground vehicle by a UAV. In *2008 American Control Conference*, pages 669–671. IEEE.

Babel, L. (2011). Trajectory planning for unmanned aerial vehicles: a network optimization approach. *Mathematical Methods of Operations Research*, 74(3):343–360.

Babel, L. (2012). Three-dimensional Route Planning for Unmanned Aerial Vehicles in a Risk Environment. *Journal of Intelligent & Robotic Systems*, 71(2):255–269.

Bae, K.-Y., Kim, Y.-D., and Han, J.-H. (2015). Finding a risk-constrained shortest path for an unmanned combat vehicle. *Computers & Industrial Engineering*, 80:245–253.

Baiocchi, V. (2014). Development of a software to optimize and plan the acquisitions from UAV and a first application in a post-seismic environment. *European Journal of Remote Sensing*, 47:477–496.

Bandeira, T. W., Coutinho, W. P., Brito, A. V., and Subramanian, A. (2015). Analysis of Path Planning Algorithms Based on Travelling Salesman Problem Embedded in UAVs. In *2015 Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 70–75.

Barraquand, J. and Latombe, J.-C. (1991). Robot Motion Planning: A Distributed Representation Approach. *The International Journal of Robotics Research*, 10(6):628–649.

Beard, R. W. and McLain, T. W. (2012). *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press.

Bednowitz, N., Batta, R., and Nagi, R. (2012). Dispatching and loitering policies for unmanned aerial vehicles under dynamically arriving multiple priority targets. *Journal of Simulation*, 8(1):9–24.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219.

Bertsekas (1979). *Stochastic Optimal Control: The Discrete Time Case*. Elsevier Science.

Besada-Portas, E., de la Torre, L., Moreno, A., and Risco-Martín, J. L. (2013). On the performance comparison of multi-objective evolutionary UAV path planners. *Information Sciences*, 238:111–125.

Besada-Portas, E., Torre, L. d. l., Cruz, J. M. d. l., and Andrés-Toro, B. d. (2010). Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios. *IEEE Transactions on Robotics*, 26(4):619–634.

Betts, J. T. (1998). Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207.

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.

Boeing (2000). Operational Use of Angle of Attack on Modern Commercial Jet Airplanes. *Aero Magazine*, (12).

Bower, G. C. (2010). *Boundary Layer Dynamic Soaring for Autonomous Aircraft: Design and Validation*. Ph.D. Thesis, Stanford University, Stanford, California.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Bravo, R. and Leiras, A. (2015). Literature Review of the Applications of UAVs in Humanitairan Relief. In *Perspectivas Globais para a Engenharia de Produção*, pages 1–15, Fortaleza-CE, Brazil.

Bryson, A. E. (1975). *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press.

Bryson, A. E. (2002). *Applied Linear Optimal Control Paperback with CD-ROM: Examples and Algorithms*. Cambridge University Press.

Bulhões, T., Subramanian, A., Erdoğan, G., and Laporte, G. (2017). The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research*.

Casbeer, D. W. and Holsapple, R. W. (2011). Column generation for a UAV assignment problem with precedence constraints. *International Journal of Robust and Nonlinear Control*, 21(12):1421–1433.

Chakrabarty, A. and Langelaan, J. W. (2011). Energy-Based Long-Range Path Planning for Soaring-Capable Unmanned Aerial Vehicles. *Journal of Guidance, Control, and Dynamics*, 34(4):1002–1015.

Chen, Y., Yu, J., Mei, Y., Zhang, S., Ai, X., and Jia, Z. (2016). Trajectory optimization of multiple quad-rotor UAVs in collaborative assembling task. *Chinese Journal of Aeronautics*, 29(1):184–201.

Choe, R., Puig-Navarro, J., Cichella, V., Xargay, E., and Hovakimyan, N. (2016). Cooperative Trajectory Generation Using Pythagorean Hodograph Bézier Curves. *Journal of Guidance, Control, and Dynamics*, 39(8):1–20.

Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S. G., and Bian, L. (2017). Drones for disaster response and relief operations: A continuous approximation model. *International Journal of Production Economics*, 188:167–184.

Cobano, J. A., Alejo, D., Sukkarieh, S., Heredia, G., and Ollero, A. (2013). Thermal detection and generation of collision-free trajectories for cooperative soaring UAVs. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2948–2954.

Colasurdo, G., Zavoli, A., Longo, A., Casalino, L., and Simeoni, F. (2014). Tour of Jupiter Galilean moons: Winning solution of GTOC6. *Acta Astronautica*, 102:190–199.

Cons, M. S., Shima, T., and Domshlak, C. (2014). Integrating Task and Motion Planning for Unmanned Aerial Vehicles. *Unmanned Systems*, 02(01):19–38.

Conway, B. A. (2010). *Spacecraft Trajectory Optimization*. Cambridge University Press.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. P., and Vigo, D. (2007). Chapter 6 Vehicle Routing. In *Handbooks in Operations Research and Management Science*, volume 14 of *Transportation*, pages 367–428. Elsevier.

Coutinho, W. P. (2017). Glider routing. `https://www.youtube.com/playlist?list=PL1mldBX67GxrUkuQZSzArTRLbYOowDD45`. Accessed: 25/05/2017.

Crispin, C. (2016). *Path Planning Algorithms for Atmospheric Science Applications of Autonomous Aircraft Systems*. Ph.D. Thesis, University of Southampton, Southampton, UK.

Cruz, F., Subramanian, A., Bruck, B. P., and Iori, M. (2017). A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. *Computers & Operations Research*, 79:19 – 33.

Darrah, M., Fuller, E., Munasinghe, T., Duling, K., Gautam, M., and Wathen, M. (2012). Using Genetic Algorithms for Tasking Teams of Raven UAVs. *Journal of Intelligent & Robotic Systems*, 70(1-4):361–371.

Deittert, M., Richards, A., Toomer, C. A., and Pipe, A. (2009). Engineless Unmanned Aerial Vehicle Propulsion by Dynamic Soaring. *Journal of Guidance, Control, and Dynamics*, 32(5):1446–1457.

Delahaye, D., Puechmorel, S., Tsiotras, P., and Feron, E. (2014). Mathematical Models for Aircraft Trajectory Design: A Survey. In Institute, E. N. R., editor, *Air Traffic Management and Systems*, number 290 in Lecture Notes in Electrical Engineering, pages 205–247. Springer Japan.

Dilão, R. and Fonseca, J. (2013). Dynamic Trajectory Control of Gliders. In Chu, Q., Mulder, B., Choukroun, D., Kampen, E.-J. v., Visser, C. d., and Looye, G., editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 373–386. Springer Berlin Heidelberg.

Drew, D. R., Barlow, J. F., and Lane, S. E. (2013). Observations of wind speed profiles over greater london, uk, using a doppler lidar. *Journal of Wind Engineering and Industrial Aerodynamics*, 121:98–105.

Edison, E. and Shima, T. (2011). Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research*, 38(1):340–356.

Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483.

Elston, J., Argrow, B., Stachura, M., Weibel, D., Lawrence, D., and Pope, D. (2014). Overview of Small Fixed-Wing Unmanned Aircraft for Meteorological Sampling. *Journal of Atmospheric and Oceanic Technology*, 32(1):97–115.

Enright, J. J., Frazzoli, E., Pavone, M., and Savla, K. (2015). UAV Routing and Coordination in Stochastic, Dynamic Environments. In Valavanis, K. P. and Vachtsevanos, G. J., editors, *Handbook of Unmanned Aerial Vehicles*, pages 2079–2109. Springer Netherlands.

Evers, L., Barros, A. I., Monsuur, H., and Wagelmans, A. (2014). Online stochastic UAV mission planning with time windows and time-sensitive targets. *European Journal of Operational Research*, 238(1):348–362.

Faied, M., Mostafa, and Girard, A. (2010). Vehicle Routing Problem Instances: Application to Multi-UAV Mission Planning. In *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences, pages 1–11. American Institute of Aeronautics and Astronautics.

Filippis, L. D., Guglieri, G., and Quagliotti, F. (2011). Path Planning Strategies for UAVS in 3d Environments. *Journal of Intelligent & Robotic Systems*, 65(1-4):247–264.

Fisch, F. (2011). *Development of a Framework for the Solution of High-Fidelity Trajectory Optimization Problems and Bilevel Optimal Control Problems.* Ph.D. Thesis, Technical University of Munich, Munich, Germany.

Flanzer, T. (2012). *Robust Trajectory Optimisation and Control of a Dynamic Soaring Unmanned Aerial Vehicle.* PhD. Thesis, Stanford University, Stanford.

Forsmo, E. J. (2012). *Optimal Path Planning for Unmanned Aerial Systems.* Msc. Thesis, Norwegian University of Science and Technology, Norway.

Fügenschuh, A. and 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`.

Furini, F., Persiani, C. A., and Toth, P. (2016). The Time Dependent Traveling Salesman Planning Problem in Controlled Airspace. *Transportation Research Part B: Methodological*, 90:38–55.

Galceran, E. and Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276.

Gao, X., Hou, Z., Guo, Z., Fan, R., and Chen, X. (2014). Analysis and design of guidance-strategy for dynamic soaring with UAVs. *Control Engineering Practice*, 32:218–226.

García-Heras, J., Soler, M., and Sáez, F. J. (2014). A Comparison of Optimal Control Methods for Minimum Fuel Cruise at Constant Altitude and Course with Fixed Arrival Time. *Procedia Engineering*, 80:231–244.

Gasparetto, A., Boscariol, P., Lanzutti, A., and Vidoni, R. (2015). Path Planning and Trajectory Planning Algorithms: A General Overview. In *Motion and Operation Planning of Robotic Systems*, Mechanisms and Machine Science, pages 3–27. Springer, Cham.

Goddard, R. H. (1919). A method of reaching extreme altitudes (with 10 plates). *Smithsonian Miscellaneous Collections*, 71(2):1–69.

Goerzen, C., Kong, Z., and Mettler, B. (2009). A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65.

Golden, B., Raghavan, S., and Wasil, E., editors (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*. Springer US, Boston, MA.

Golden, B. L. and Assad, A. (1988). *Vehicle Routing: Methods and Studies*. Amsterdam: North-Holland.

Gottlieb, Y. and Shima, T. (2015). UAVs Task and Motion Planning in the Presence of Obstacles and Prioritized Targets. *Sensors*, 15(11):29734–29764.

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.

Guerriero, F., Surace, R., Loscrí, V., and Natalizio, E. (2014). A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. *Applied Mathematical Modelling*, 38(3):839–852.

Ha, Q. M., Deville, Y., Pham, Q. D., and Há, M. H. (2015). On the Min-cost Traveling Salesman Problem with Drone. Working Paper.

Hairer, E., Nørsett, S. P., and Wanner, G. (2011). *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 1 of *Springer Series in Computational Mathematics*. Springer.

Hajiyev, C., Soken, H. E., and Vural, S. Y. (2015). Equations of Motion for an Unmanned Aerial Vehicle. In *State Estimation and Control for Low-cost Unmanned Aerial Vehicles*, pages 9–23. Springer International Publishing.

Han, D.-H., Kim, Y.-D., and Lee, J.-Y. (2014). Multiple-criterion shortest path algorithms for global path planning of unmanned combat vehicles. *Computers & Industrial Engineering*, 71:57–69.

Harris, M. W. and Acikmese (2013). Maximum Divert for Planetary Landing Using Convex Optimization. *Journal of Optimization Theory and Applications*, 162(3):975–995.

Hayat, S., Yanmaz, E., and Muzaffar, R. (2016). Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Communications Surveys & Tutorials*, 18:1–1.

Henchey, M. J., Batta, R., Karwan, M., and Crassidis, A. (2016). A Flight Time Approximation Model for Unmanned Aerial Vehicles. In (Ret.), US Navy, J. R. C. C. and Jr, J. Q. D., editors, *Operations Research for Unmanned Systems*, pages 95–117. John Wiley & Sons, Ltd.

Ho, H.-M. and Ouaknine, J. (2015). The Cyclic-Routing UAV Problem is PSPACE-Complete. In Pitts, A., editor, *Foundations of Software Science and Computation Structures*, number 9034 in Lecture Notes in Computer Science, pages 328–342. Springer Berlin Heidelberg.

How, J. P., Frazzoli, E., and Chowdhary, G. V. (2015). Linear Flight Control Techniques for Unmanned Aerial Vehicles. In Valavanis, K. P. and Vachtsevanos, G. J., editors, *Handbook of Unmanned Aerial Vehicles*, pages 529–576. Springer Netherlands.

Hu, X., Cheng, J., and Luo, H. (2015a). Task Assignment for Multi-UAV under Severe Uncertainty by Using Stochastic Multicriteria Acceptability Analysis. *Mathematical Problems in Engineering*, 2015:1–10.

Hu, X., Ma, H., Ye, Q., and Luo, H. (2015b). Hierarchical method of task assignment for multiple cooperating UAV teams. *Journal of Systems Engineering and Electronics*, 26(5):1000–1009.

Huang, L., Qu, H., Ji, P., Liu, X., and Fan, Z. (2016). A novel coordinated path planning method using k-degree smoothing for multi-UAVs. *Applied Soft Computing*, 48:182–192.

Jaishankar, S. and Pralhad, R. N. (2011). 3d Off-Line Path Planning For Aerial Vehicle Using Distance Transform Technique. *Procedia Computer Science*, 4:1306–1315.

Jiang, J. and Ng, K. M. (2011). Priority-based routing of unmanned combat aerial vehicles. In *Defense Science Research Conference and Expo (DSR), 2011*, pages 1–4.

Kagabo, W. (2010). *Optimal trajectory planning for a UAV glider using atmospheric thermals*. Msc. Thesis, Rochester Institute of Technology, Rochester, New York.

Kanistras, K., Martins, G., Rutherford, M., and Valavanis, K. (2013). A survey of unmanned aerial vehicles (uavs) for traffic monitoring. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 221–234.

Karaman, S. and Inalhan, G. (2008). Large-scale Task/Target Assignment for UAV Fleets Using a Distributed Branch and Price Optimization Scheme. In *Proceedings of the 17th World Congress*, volume 41 of *17th IFAC World Congress*, pages 13310–13317.

Keane, A., Scanlan, J., Lock, A., Ferraro, M., Spillane, P., and Breen, J. (2017). Maritime flight trials of the southampton university laser sintered aircraft: Project albatross. *The Aeronautical Journal of the Royal Aeronautical Society*, page to appear. Available at `https://eprints.soton.ac.uk/411713/`.

Keane, J. F. and Carr, S. S. (2013). A Brief History of Early Unmanned Aircraft. *Johns Hopkins Apl Technical Digest*, 32(3):558–571.

Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., and Balas, G. (2008). Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations. *IEEE Transactions on Control Systems Technology*, 16(1):19–33.

Khamis, A., Hussein, A., and Elmogy, A. (2015). Multi-robot Task Allocation: A Review of the State-of-the-Art. In Koubâa, A. and Dios, J. R. M.-d., editors, *Cooperative Robots and Sensor Networks 2015*, number 604 in Studies in Computational Intelligence, pages 31–51. Springer International Publishing.

Kim, M.-H., Baik, H., and Lee, S. (2015). Resource Welfare Based Task Allocation for UAV Team with Resource Constraints. *Journal of Intelligent & Robotic Systems*, 77(3-4):611–627.

Kim, Y., Gu, D.-W., and Postlethwaite, I. (2007). Real-Time Optimal Time-Critical Target Assignment for UAVs. In Pardalos, P. M., Murphey, R., Grundel, D., and Hirsch, M. J., editors, *Advances in Cooperative Control and Optimization*, number 369 in Lecture Notes in Control and Information Sciences, pages 265–280. Springer Berlin Heidelberg.

Kingston, D. B. and Schumacher, C. J. (2005). Time-dependent cooperative assignment. In *ACC: Proceedings of the 2005 American Control Conference, Vols 1-7*, pages 4084–4089. IEEE, New York.

Kirk, D. E. (2012). *Optimal Control Theory: An Introduction*. Courier Corporation.

Kivelevitch, E., Cohen, K., and Kumar, M. (2016). Near-Optimal Assignment of UAVs to Targets Using a Market-Based Approach. In (Ret.), US Navy, J. R. C. C. and Jr, J. Q. D., editors, *Operations Research for Unmanned Systems*, pages 27–57. John Wiley & Sons, Ltd.

Kramer, R., Subramanian, A., Vidal, T., and dos Anjos F. Cabral, L. (2015). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, 243(2):523 – 539.

Kroo, I. (2001). *Aircraft Design: Synthesis and Analysis*. Desktop Aeronautics, P.O. Box 20384, Stanford, CA 94309. Version 0.99.

Kumar, P. and Padhi, R. (2013). Suboptimal Guidance of UAVs Satisfying Multiple Constraints on Waypoints. *IFAC Proceedings Volumes*, 46(19):78–83.

Kunchev, V., Jain, L., Ivancevic, V., and Finn, A. (2006). Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. In Gabrys, B., Howlett, R. J., and Jain, L. C., editors, *Knowledge-Based Intelligent Information and Engineering Systems*, number 4252 in Lecture Notes in Computer Science, pages 537–544. Springer Berlin Heidelberg.

Kwak, D. J., Moon, S., Kim, S., and Kim, H. J. (2013). Optimization of Decentralized Task Assignment for Heterogeneous UAVs. In *Proceedings of the 11th IFAC Workshop on Adaptation and Learning in Control and Signal Processing*, volume 46, pages 251–256.

Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14.

Langelaan, J. (2007). Long Distance/Duration trajectory optimization for small UAVs. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics.

Latombe, J.-C. (1991). Introduction and Overview. In *Robot Motion Planning*, number 124 in The Springer International Series in Engineering and Computer Science, pages 1–57. Springer US.

Laville, S., Ross, A., Topping, A., Gayle, D., and Grierson, J. (2017, June 15). Grenfell tower: firefighters search overnight with toll expected to rise. `https://www.theguardian.com/uk-news/2017/jun/14/fire-24-storey-grenfell-tower-block-white-city-latimer-road-london`.

Levy, D., Sundar, K., and Rathinam, S. (2014). Heuristics for Routing Heterogeneous Unmanned Vehicles with Fuel Constraints. *Mathematical Problems in Engineering*, 2014:1–12.

Lin, L., Qibo, S., Shangguang, W., and Fangchun, Y. (2013). Research on PSO Based Multiple UAVs Real-Time Task Assignment. In *2013 25th Chinese Control and Decision Conference*, pages 1530–1536. IEEE, New York.

Liu, H., Lin, M., and Deng, L. (2016). UAV route planning for aerial photography under interval uncertainties. *Optik - International Journal for Light and Electron Optics*, 127(20):9695–9700.

Liu, W., Zheng, Z., and Cai, K.-Y. (2013). Bi-level programming based real-time path planning for unmanned aerial vehicles. *Knowledge-Based Systems*, 44:34–47.

Manyam, S. G., Rathinam, S., and Darbha, S. (2015). Computation of Lower Bounds for a Multiple Depot, Multiple Vehicle Routing Problem With Motion Constraints. *Journal of Dynamic Systems, Measurement, and Control*, 137(9):1–6.

Maolaaisha, A. (2015). *Free-Flight Trajectory Optimization by Mixed Integer Programming*. Master Thesis, University of Hamburg, Hamburg.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, 40(8):2145–2160.

Medeiros, A. C. and Urrutia, S. (2010). Discrete optimization methods to determine trajectories for Dubins' vehicles. *Electronic Notes in Discrete Mathematics*, 36:17–24.

Mersheeva, V. (2015). *UAV Routing Problem for Area Monitoring in a Disaster Situation*. Ph.D. Thesis, Alpen-Adria-Universität Klagenfurt, Austria.

Mittelmann, H. D. (2017). Mixed-integer socp benchmark. `http://plato.asu.edu/ftp/misocp.html`. Accessed: 14/07/2017.

Mufalli, F., Batta, R., and Nagi, R. (2012). Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans. *Computers & Operations Research*, 39(11):2787–2799.

Murray, C. and Karwan, M. (2013). A branch-and-bound-based solution approach for dynamic rerouting of airborne platforms. *Naval Research Logistics (NRL)*, 60(2):141–159.

Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.

Murray, C. C. and Karwan, M. H. (2010). An extensible modeling framework for dynamic reassignment and rerouting in cooperative airborne operations. *Naval Research Logistics (NRL)*, 57(7):634–652.

Myers, D., Batta, R., and Karwan, M. (2016). A real-time network approach for including obstacles and flight dynamics in UAV route planning. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 13(3):291–306.

Nedjati, A., Izbirak, G., Vizvari, B., and Arkat, J. (2016a). Complete coverage path planning for a multi-uav response system in post-earthquake assessment. *Robotics*, 5(4).

Nedjati, A., Vizvari, B., and Izbirak, G. (2016b). Post-earthquake response by small UAV helicopters. *Natural Hazards*, 80(3):1669–1688.

Nex, F. and Remondino, F. (2013). UAV for 3d mapping applications: a review. *Applied Geomatics*, 6(1):1–15.

Nguyen, J. L., Lawrance, N. R. J., Fitch, R., and Sukkarieh, S. (2015). Real-time path planning for long-term information gathering with an aerial glider. *Autonomous Robots*, 40(6):1–23.

Niccolini, M., Innocenti, M., and Pollini, L. (2010). Multiple UAV Task Assignment using Descriptor Functions. *IFAC Proceedings Volumes*, 43(15):93–98.

NPD (2016). Year-Over-Year Drone Revenue Soars, According to NPD. `https://www.npd.com/wps/portal/npd/us/news/press-releases/2016/year-over-year-drone-revenue-soars-according-to-npd/`. Accessed: 15-10-2016.

Park, C.-H., Kim, Y.-D., and Jeong, B. (2012). Heuristics for determining a patrol path of an unmanned combat vehicle. *Computers & Industrial Engineering*, 63(1):150–160.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

Penna, P. H. V., Subramanian, A., and Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19:201–232.

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., and Prins, C. (2017). A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research*, pages 1–70.

Pepy, R. and Hérissé, B. (2014). An Indirect Method for Optimal Guidance of a Glider. *IFAC Proceedings Volumes*, 47(3):5097–5102.

Pharpatara, P., Hérissé, B., and Bestaoui, Y. (2015). 3D-shortest paths for a hypersonic glider in a heterogeneous environment. *IFAC-PapersOnLine*, 48(9):186–191.

Quaritsch, M., Kruggl, K., Wischounig-Strucl, D., Bhattacharya, S., Shah, M., and Rinner, B. (2010). Networked UAVs as aerial sensor network for disaster management applications. *Elektrotech. Inftech.*, 127(3):56–63.

Raivio, T., Ehtamo, H., and Hämäläinem, R. P. (1996). Aircraft trajectory optimization using nonlinear programming. In Dolezăl, J. and Fidler, J., editors, *System Modelling and Optimization*, IFIP: The International Federation for Information Processing, pages 435–441. Springer US.

Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M. D., and Camacho, D. (2016). Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. *Soft Computing*, To appear:1–18.

Rao, A. V. (2014). Trajectory Optimization: A Survey. In Waschl, H., Kolmanovsky, I., Steinbuch, M., and Re, L. d., editors, *Optimization and Optimal Control in Automotive Systems*, number 455 in Lecture Notes in Control and Information Sciences, pages 3–21. Springer International Publishing.

Rao, B., Gopi, A. G., and Maione, R. (2016). The societal impact of commercial drones. *Technology in Society*, 45:83–90.

Rathinam, S. and Sengupta, R. (2007). Algorithms for Routing Problems Involving UAVs. In Chahl, D. J. S., Jain, P. L. C., Mizutani, D. A., and Sato-Ilic, P. M., editors, *Innovations in Intelligent Machines - 1*, number 70 in Studies in Computational Intelligence, pages 147–172. Springer Berlin Heidelberg.

Rayleigh, L. (1883). The Soaring of Birds. *Nature*, 27(701):534–535.

Reif, J. and Sharir, M. (1994). Motion Planning in the Presence of Moving Obstacles. *J. ACM*, 41(4):764–790.

Roelofsen, S., Martinoli, A., and 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)*, pages 2555–2560.

Rogowski, K. and Maroński, R. (2011). Optimization of glider's trajectory for given thermal conditions. *Archive of Mechanical Engineering*, 58(1):11–25.

Ropke, S. and Pisinger, D. (2006). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 171(3):750–775.

Ross, I. M. (2009). *A Primer on Pontryagin's Principle in Optimal Control*. Collegiate Publishers.

Russell, J. (1996). *Performance and Stability of Aircraft*. Butterworth-Heinemann.

Ruzgienė, B., Berteška, T., Gečyte, S., Jakubauskienė, E., and Aksamitauskas, V. Č. (2015). The surface modelling based on UAV Photogrammetry and qualitative estimation. *Measurement*, 73:619–627.

Rysdyk, R. (2006). Unmanned Aerial Vehicle Path Following for Target Observation in Wind. *Journal of Guidance, Control, and Dynamics*, 29(5):1092–1100.

Sartorius, S. (2013). Oswald efficiency estimation function. `https://uk.mathworks.com/matlabcentral/fileexchange/38800-oswald-efficiency-estimation-function?focused=3795877&tab=function`.

Schultz, R. L. and Zagalsky, N. R. (1972). Aircraft performance optimization. *Journal of Aircraft*, 9(2):108–114.

Shanmugavel, M. (2013). Dynamic risk uncertainties by cooperative sites in path planning of UAVs. *IFAC Proceedings Volumes*, 46(30):310–315.

Shaw-Cortez, W. E. and Frew, E. (2015). Efficient Trajectory Development for Small Unmanned Aircraft Dynamic Soaring Applications. *Journal of Guidance, Control, and Dynamics*, 38(3):519–523.

Shima, T. and Schumacher, C. (2009). Assigning cooperating UAVs to simultaneous tasks on consecutive targets using genetic algorithms. *Journal of the Operational Research Society*, 60(7):973–982.

Silva, M. M., Subramanian, A., Vidal, T., and Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research*, 221(3):513–520.

Silva, W., Frew, E. W., and Shaw-Cortez, W. (2015). Implementing path planning and guidance layers for dynamic soaring and persistence missions. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 92–101.

Sobester, A., Castro, I. P., Czerski, H., and Zapponi, N. (2013). Notes on Meteorological Balloon Mission Planning. In *AIAA Balloon Systems (BAL) Conference*, pages 1–15. American Institute of Aeronautics and Astronautics.

Soler, M., Zou, B., and Hansen, M. (2014). Flight trajectory design in the presence of contrails: Application of a multiphase mixed-integer optimal control approach. *Transportation Research Part C: Emerging Technologies*, 48:172–194.

Song, B. D., Kim, J., and Morrison, J. R. (2016). Rolling horizon path planning of an autonomous system of uavs for persistent cooperative service: MILP formulation and efficient heuristics. *Journal of Intelligent & Robotic Systems*, 84(1):241–258.

Stengel, R. F. (2004). *Flight Dynamics*. Princeton University Press.

Stöcker, C., Eltner, A., and Karrasch, P. (2015). Measuring gullies by synergetic application of UAV and close range photogrammetry: A case study from Andalusia, Spain. *Catena*, 132:1–11.

Stryk, O. v. and Bulirsch, R. (1992). Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373.

Stump, E. and Michael, N. (2011). Multi-robot persistent surveillance planning as a Vehicle Routing Problem. In *2011 IEEE Conference on Automation Science and Engineering (CASE)*, pages 569–575.

Subramanian, A. (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. Ph.D. Thesis, Universidade Federal Fluminense, Niterói.

Subramanian, A. and Battarra, M. (2013). An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. *Journal of the Operational Research Society*, 64(3):402–409.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531.

Sundar, K. and Rathinam, S. (2014). Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots. *IEEE Transactions on Automation Science and Engineering*, 11(1):287–294.

Techy, L., Woolsey, C. A., and Morgansen, K. A. (2010). Planar path planning for flight vehicles in wind with turn rate and acceleration bounds. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3240–3245.

Thi, H. A. L., Nguyen, D. M., and Dinh, T. P. (2012). Globally solving a nonlinear UAV task assignment problem by stochastic and deterministic optimization approaches. *Optimization Letters*, 6(2):315–329.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics.

Tsourdos, A., White, B., and Shanmugavel, M. (2010). *Cooperative Path Planning of Unmanned Aerial Vehicles*. John Wiley & Sons, Ltd.

Uysal, M., Toprak, A. S., and Polat, N. (2015). DEM generation with UAV Photogrammetry and accuracy analysis in Sahitler hill. *Measurement*, 73:539–543.

Vanderbei, R. J. (2001). Case studies in trajectory optimization: Trains, planes, and other pastimes. *Optimization and Engineering*, 2(2):215–243.

Vidal, T., Battarra, M., Subramanian, A., and Erdoğan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research.*, 58:87 – 99.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, 60(3):611–624.

Vilar, R. G. and Shin, H.-S. (2013). Communication-Aware Task Assignment for UAV Cooperation in Urban Environments. In *Proceedings of the 2nd IFAC Workshop on Research, Education and Development of Unmanned Aerial Systems*, volume 46, pages 352–359.

Wang, G.-G., Chu, H. E., and Mirjalili, S. (2016). Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerospace Science and Technology*, 49:231–238.

Wang, J., Zhang, Y. F., Geng, L., Fuh, J. Y. H., and Teo, S. H. (2015). A Heuristic Mission Planning Algorithm for Heterogeneous Tasks with Heterogeneous UAVs. *Unmanned Systems*, 03:205–219.

Wang, X. (2009). Solving optimal control problems with MATLAB: Indirect methods. Technical report, ISE. Dept., NCSU, Raleigh, NC 27695.

Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, 11(4):679–697.

Wu, H., Chio Cho, N., Bouadi, H., Zhong, L., and Mora-Camino, F. (2012). Dynamic programming for trajectory optimization of engine-out transportation aircraft. In *Control and Decision Conference (CCDC), 2012 24th Chinese*, pages 98–103.

Wu, J.-P., Peng, Z.-H., and Chen, J. (2011). 3d Multi-Constraint Route Planning for UAV Low-altitude Penetration Based on Multi-Agent Genetic Algorithm. *IFAC Proceedings Volumes*, 44(1):11821–11826.

Xie, F., Potts, C. N., and Bektaş, T. (2017). Iterated local search for workforce scheduling and routing problems. *Journal of Heuristics*, 23(6):471–500.

Xu, S., Doğançay, K., and Hmam, H. (2017). Distributed pseudolinear estimation and UAV path optimization for 3d AOA target tracking. *Signal Processing*, 133:64–78.

Xu, Z., Yang, J., Peng, C., Wu, Y., Jiang, X., Li, R., Zheng, Y., Gao, Y., Liu, S., and Tian, B. (2014). Development of an UAS for post-earthquake disaster surveying and its application in ms7.0 lushan earthquake, sichuan, china. *Computers & Geosciences*, 68:22–30.

Yakıcı, E. (2016). Solving Location and Routing Problem for UAVs. *Computers & Industrial Engineering*, 102:294–301.

Yang, L., Qi, J., Song, D., Xiao, J., Han, J., and Xia, Y. (2016). Survey of Robot 3d Path Planning Algorithms. *Journal of Control Science and Engineering*, 2016:1–22.

Yang, Y., Karimadini, M., Xiang, C., Teo, S. H., Chen, B. M., and Lee, T. H. (2015). Wide area surveillance of urban environments using multiple Mini-VTOL UAVs. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 795–800.

Yomchinda, T., Horn, J. F., and Langelaan, J. W. (2016). Modified Dubins parameterization for aircraft emergency trajectory planning. *Journal of Aerospace Engineering*, 1:1–20.

YongBo, C., YueSong, M., JianQiao, Y., XiaoLong, S., and Nuo, X. (2017). Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing*, 266(Supplement C):445 – 457.

Yuan, C., Zhang, Y., and Liu, Z. (2015). A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Can. J. For. Res.*, 45(7):783–792.

Zapponi, N. (2013). Astra high altitude balloon flight planner. `http://astra-planner.soton.ac.uk/`. Accessed: 21/02/2017.

Zhang, X., Chen, J., Xin, B., and Fang, H. (2011). Online Path Planning for UAV Using an Improved Differential Evolution Algorithm. *IFAC Proceedings Volumes*, 44(1):6349–6354.

Zhang, X., Chen, J., Xin, B., and Peng, Z. (2014). A memetic algorithm for path planning of curvature-constrained UAVs performing surveillance of multiple ground targets. *Chinese Journal of Aeronautics*, 27(3):622–633.

Zhang, Y., Chen, J., and 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.

Zhou, F., Li, X., and Ma, J. (2017). Parsimonious shooting heuristic for trajectory design of connected automated traffic part i: Theoretical analysis with generalized time geography. *Transportation Research Part B: Methodological*, 95:394–420.