

Waypoint-Free Trajectory Planning for Sequencing Airport Arrivals

Maël Feurgard ^{*}, Nicolas Durand , and Gautier Hattenberger 

École Nationale de l'Aviation Civile, Toulouse, France

*Corresponding author: mael.feurgard@enac.fr

Abstract

The Terminal Maneuvering Area of airports are major bottlenecks, as incoming traffic must be sequenced to meet schedules while ensuring safe separation distances within a limited space. Most of the traffic is handled manually, relying on experience and airport-dependent procedures with varying levels of efficiency.

We present the Dubins Fleet Planner, a new approach to optimize flight paths in air traffic. This path planning method was originally developed for uncrewed aerial systems, and we adapt it to the context of commercial air traffic. By taking into account the characteristics of each aircraft and its possible arrival times, this planner is capable of generating horizontally separated flight paths, consisting solely of straight lines and circular turns, performed at a constant speed and minimizing the cumulative flight time. This method is highly experimental, as it does not follow the usual navigation beacons.

We benchmarked the method against traffic measured at the London Heathrow Airport for a full day. 1280 flights were considered, and only 3 had to be removed for loss of separation. On average, arriving flights were delayed by 86 seconds, while departing flights reached separation altitude 79 seconds ahead of schedule. The full day was simulated in 737 seconds, with an average planning time of 3 seconds per case. Data was provided by the OpenSky historical database.

Keywords: Aircraft Sequencing Problem; Air traffic management; Trajectory based operation; Combinatorial optimization

Abbreviations: TMA: Terminal Maneuvering Area, DFP: Dubins Fleet Planner, ATCO: Air Traffic Management, ILP: Integer Linear Programming, NM: Nautical mile, kt: knots (speed)

1. Introduction

The Terminal Maneuvering Areas (TMAs) of airports are major bottlenecks, as incoming traffic must be sequenced to meet schedules while ensuring safe separation distances within a limited space. Most of the traffic is handled manually, relying on expertise and airport-dependent procedures with varying levels of efficiency [1]. Among existing procedures, the Point Merge [2] is becoming very popular because it is compliant with human management. This strategy involves positioning the aircraft along a single arc centered on the approach point. By choosing a large radius, several aircraft can be placed on hold along the arc by having them circle in the same direction. Since they all maintain the same distance from the approach point, the aircraft can be sequenced simply by calling them in the order in which they should land. However, it requires significant airspace, and does not optimize trajectories.

Several studies have explored ways to improve throughput based on optimization techniques. According Ikli et al. [3], most of them optimize throughput only at the runway level. Some consider the full trajectories: Liang et al. [4] handle multiple point merge structures, Gui et al. [5] introduce deviations between waypoints to increase the number of possible paths.

Path planning in the context of Air Traffic Control (ATC) has to take into account many constraints. Indeed, ATC Operators (ATCOs) such as pilots and air traffic controllers should remain involved in the loop, and the trajectories should be easy for them to read and follow. Moreover, with continuous descent approaches becoming environmentally more acceptable, but stressful to manage by ATCO, efficient horizontal separation between aircraft could become strategically important.

We use the Dubins Fleet Planner (DFP) to plan the paths of aircraft landing at and departing from a given airport. The DFP was initially developed to generate coordinated collision-free trajectories for fixed-wing uncrewed aircraft [6]. The generated trajectories assume a constant airspeed and are made of line segments and circular arcs, which make them comparable to the trajectories currently used in ATC. The DFP ensures separation through horizontal distance, thus not interfering with continuous descent operations. And finally, its requirement for synchronous arrivals can be adapted to ensure that flights land on time.

We first present the Dubins Fleet Planner, its core principles and how to adapt it to the ATC constraints. Then we validate our approach using actual air traffic data at the London Heathrow Airport.

2. Problem formulation

The path planning problem we are considering consists of finding flight paths for multiple aircraft flying in shared airspace, allowing them to reach their respective destinations without creating conflicts — that is, by maintaining a minimum distance between each pair of aircraft.

We are interested in ensuring separation by relying only on horizontal separation. Considering $N > 0$ aircraft to be planned, we denote for each $k \in [1, N]$ the 2D position $[x_k, y_k] \in \mathbb{R}^2$ and 2D orientation $\theta_k \in \mathbb{S} := [0, 2\pi]$. They are expressed using a common flat ground coordinate system, valid in the vicinity of the airport. Position and orientation are combined in the pose $\vec{\mathbf{p}} := [x_k, y_k, \theta_k]$. We assume simple dynamics. We consider the aircraft to always be aligned with its speed vector, and the speed magnitude to be constant, of value noted $v_k > 0$. It can turn at a limited rate, this limited being represented by the minimal turn radius $\rho_k > 0$, radius of the circle achieved when turning at maximal rate. This yields the so-called Dubins model, with equations:

$$\frac{d\vec{\mathbf{p}}_k}{dt} = \begin{pmatrix} \dot{x}_k \\ \dot{y}_k \\ \dot{\theta}_k \end{pmatrix} := \begin{pmatrix} v_k \cos \theta_k \\ v_k \sin \theta_k \\ u_k \end{pmatrix} \quad (1)$$

The sole control parameter, the change in orientation is $u_k \in [-v_k/\rho_k, v_k/\rho_k]$. Note that we can deduce θ_k from $\dot{x}_k, \dot{y}_k, v_k$. As such, for a differentiable trajectory $\gamma : \mathbb{R} \rightarrow \mathbb{R}^2$, we may abuse the notation $\vec{\mathbf{p}}_k(t) = \gamma(t)$ to state that $\gamma(t) = [x_k(t), y_k(t)]$ and $\dot{\gamma}(t) = [v_k \cos \theta_k(t), v_k \sin \theta_k(t)]$, i.e. the aircraft follows γ both in position and speed.

This model is practical since path planning is well-understood for it. Furthermore, the constant speed assumption is reasonable and desirable for approaching aircraft.

We now introduce the multi-aircraft path planning optimization problem. Let $\vec{S}_k, \vec{E}_k \in \mathbb{R}^2 \times \mathbb{S}$ be the initial and final poses for aircraft $k \in [1, N]$. Let $\delta > 0$ the minimal horizontal separation required between any two aircraft, and let $\mathcal{O}(t) \subset \mathbb{R}^2$ be the set of forbidden airspace at a given time t . This time-varying set represents the limits of the airport TMA as well as other aircraft which are not currently being planned but are still to be avoided. The objective is to find for each aircraft k a valid

trajectory γ_k of duration $\tau_k > 0$ such that we minimize the sum of flight times. Formally, it is written:

$$\begin{aligned}
 & \underset{(\gamma_k), (\tau_k) \in \mathbb{R}^+, 1 \leq k \leq N}{\text{minimize}} && \sum_{1 \leq k \leq N} \tau_k && \text{(Cumulated flight time)} \\
 & \text{subject to} && && \\
 & \forall k \begin{cases} \gamma_k(0) &= \vec{S}_k \\ \gamma_k(\tau_k) &= \vec{E}_k \end{cases} && \text{(Correct endpoints),} \\
 & \forall k \forall t \in [0, \tau_k] \|\dot{\gamma}_k(t)\| = v_k && \text{(Constant speed),} \\
 & \forall k \forall t \in [0, \tau_k] \|\ddot{\gamma}_k(t)\| \leq \frac{v_k^2}{\rho_k} && \text{(Bounded acceleration),} \\
 & \forall k \forall t \in [0, \tau_k] \gamma_k(t) \notin \mathbb{O}(t) && \text{(Obstacles avoidance),} \\
 & \forall k_1, k_2 \forall t \in [0, \min(\tau_{k_1}, \tau_{k_2})] \|\gamma_{k_1}(t) - \gamma_{k_2}(t)\| > \delta && \text{(Minimal separation)}
 \end{aligned} \tag{2}$$

This problem is usually solved by defining a family of curves Γ in which we look for satisfying solutions. Popular families are sequences of segments or splines, but they both have their shortcomings. Sequences of segments are not differentiable, and do not represent well the turns. On the other hand, splines have many degrees of freedom, making optimization difficult, and yielding trajectories which are usually not human-readable. Thus we propose to build Γ out of Dubins paths, made of straight segments and circles arcs, ensuring both speed continuity and human readability.

3. The Dubins Fleet Planner

We first introduced the Dubins Fleet Planner in [6] to solve path planning for a fixed-wing aircraft fleet with simultaneous arrivals. This algorithm can be summarized as follows: define a travel time for the fleet, generate multiple paths per aircraft that satisfy the given travel time, and search a conflict-free combination of paths. If no valid combination of paths is found, this process is repeated with a different travel time until a solution is found or the maximum execution time is reached.

A key principle of the Dubins Fleet Planner is that the travel time is known before paths are generated. This limits the set of possible trajectories, making the search for a feasible combination of paths computationally tractable. Here, travel times are independent of one aircraft to another. In order to keep using the same path-generation techniques, we discretize the set of possible flight times for each aircraft based on its scheduled landing time.

In the following subsections, we describe how to generate Dubins paths fitted to a given travel time compatible with ATC principles, then how to detect eventual conflicts between two such paths, and finally how this comes together into a path planning solver suited for problem 2.

3.1 Fitted Dubins paths

Dubins showed in [7] that the shortest path between any two states of a vehicle described in 1 is made of straight lines and circle arcs of radius ρ_k . We denote the possible components S, L, R , respectively for Straight, Left and Right [turn]. The optimal solution is among the following six possibilities: RSR, LSL ; RSL, LSR and LRL, RLR .

We may still need more variety in the choice of paths in order to avoid conflicts. We add the single-turn paths SLS and SRS . They have a limited scope of application, but can still extend the number of possibilities. With the six basic Dubins paths and these two, we define the set of *basic paths* Γ_0 , which has cardinality 8. For any type $\gamma \in \Gamma_0$, a path is exactly defined by starting and ending poses $\vec{S}, \vec{E} \in \mathbb{R}^2 \times \mathbb{S}$ and a turn radius $\rho > 0$, and we denote $\gamma[\vec{S}, \vec{E}, \rho] \in \mathcal{D}^2(\mathbb{R}, \mathbb{R}^2)$ the corresponding

function, when it exists. By construction, all of these solutions are guaranteed to satisfy the Constant speed and Bounded acceleration constraints.

We want to generate more solutions based on Dubins paths by increasing the travel time while keeping speed unchanged. Because speed is constant along the path, the travel time is directly linked to the path length. It is more convenient to work with the lengths of Dubins paths rather than their durations, since these paths are defined geometrically. The question of deforming Dubins paths to achieve a desired path length has already been studied, and several methods devised: artificially increase the turning radius [8], use circles with different radii [9], introduce straight extensions [10] or deform straight parts [11]. We avoid methods which deform circles for compatibility with ATC procedures, namely the use of standard rate turns. Hence we rely on a single strategy: adding straight segments at the start or end of the path.

More formally, let $\vec{S} = [x_S, y_S, \theta_S]$ and $\vec{E} = [x_E, y_E, \theta_E]$ be the respective start and end poses, and let $\ell > 0$ be the desired length. We define the initial direction $\vec{V}_S = [\cos \theta_S, \sin \theta_S, 0]$ and final one $\vec{V}_E = [\cos \theta_E, \sin \theta_E, 0]$, as well as the straight distribution ratio $r \in [0, 1]$. The optimization problem linked to fitting the path using additional straight segments is defined by:

$$\min_{l \geq 0} (\text{len}(\gamma[\vec{S} + l \cdot r \vec{V}_S, \vec{E} - l \cdot (1 - r) \vec{V}_E, \rho_{\min}]) + l - \ell)^2 \quad (3)$$

In this formulation, l is the total length added to the path, with $l \cdot r$ being the straight segment's length and $l \cdot (1 - r)$ the end segment's length. This problem amounts to root finding in one dimension, and it can be efficiently solved using already existing methods, such as Brent's method [12, Chapter 4].

The ratio parameter allows many variations, but for the sake of simplicity we use only three values: start extended ($r = 1$), noted S-AAA; end extended ($r = 0$) AAA-S and both extended ($r = 1/2$) S-AAA-S, where in all cases AAA represents one of the three letters combination characterizing an element of Γ_0 . This yields our set of paths for a given target length, denoted $\Gamma[\vec{S}, \vec{E}, \rho, \ell]$, which contains at most 3×8 paths. We showcase both the basic Dubins paths and the different straight extensions in fig. 1.

One key question remains: how to check for conflicts between paths?

3.2 Conflicts detection

Dubins paths are made of straight lines and circle arcs. Thus the question of detecting a conflict between two paths can be broken down into finding conflicts between their components. Three subcases emerge: line-line, line-circle and circle-circle. Both 2D lines and circles can be conveniently represented using complex functions.

For two parametric functions $f_1, f_2 : [0, \tau] \rightarrow \mathbb{C}$ representing two trajectories, we define two types of separations with respect to some minimal distance $\delta > 0$:

- *Temporal separation*: two points following prescribed trajectories must stay separated by at least δ .

$$\min_{t \in [0, \tau]} |f_1(t) - f_2(t)| > \delta \quad (4)$$

- *Spatial separation*: two shapes must be separated by at least δ .

$$\min_{(t_1, t_2) \in [0, \tau]^2} |f_1(t_1) - f_2(t_2)| > \delta \quad (5)$$

One can easily verify that if 5 is satisfied, then so is 4. We illustrate the two separations metrics and the division into subcases in fig. 2.

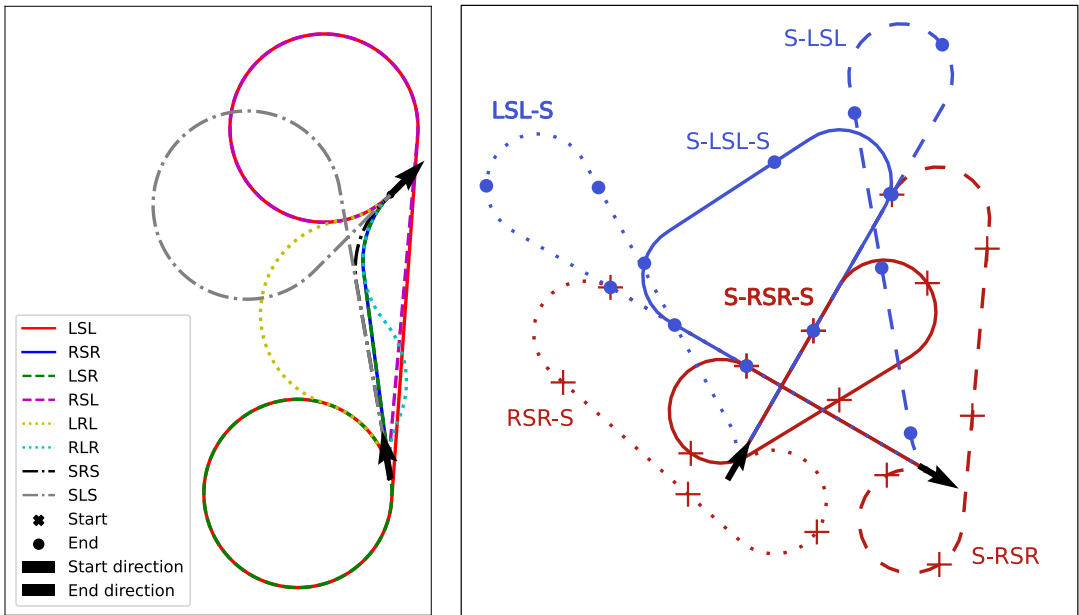


Figure 1. On the left: Examples of the six Dubins paths plus two extras (SRS, SLS)
On the right: Start, end and both extended paths for *RSR* and *LSL*.

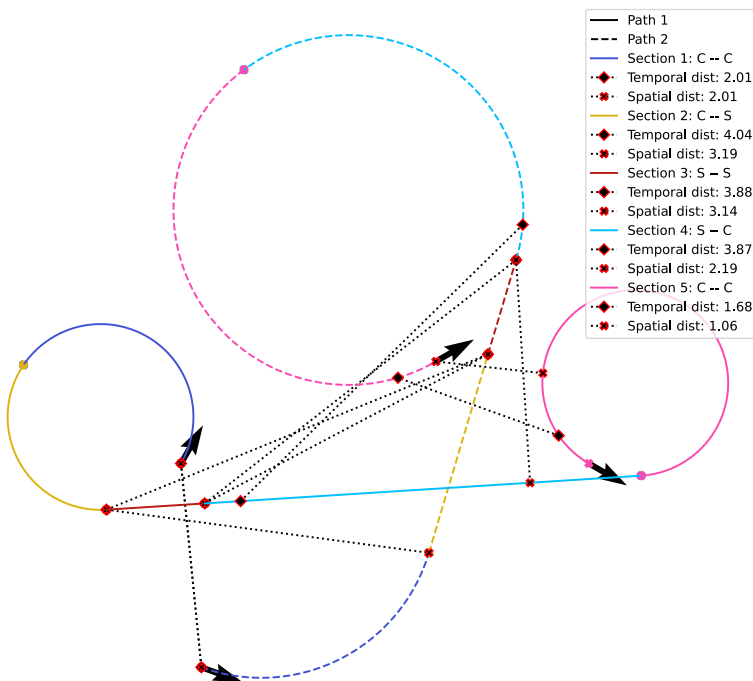


Figure 2. Distance computations between two Dubins paths, split by components.

Practically, we are interested in *temporal separation*. It is easy to compute in the line-line case, as it becomes a second degree polynomial equation. But there are no such simplifications for the others. We need a global solver, for instance the interval method [13]. We reduce the need to use such a computationally intensive method by performing pre-filtering based on *spatial separation*. This filtering is straightforward given the geometric nature of the paths considered. The geometric methods can also be used to detect conflict with static obstacles, such as the borders of the airport TMA or other forbidden flight zones.

Now that path generation and conflict detection are established, we describe the path planning algorithm.

3.3 Optimal conflict-free path allocation

Since we can generate a finite set of paths per travel time, and discretized the set of possible travel times, we can formulate the path planning problem as an optimal allocation problem. The task is defined as choosing for each aircraft a path such that no conflicts arise and minimizing total travel time.

Formally, this can be described as an Integer Linear Programming (ILP) problem. Let $N > 0$ be the number of aircraft to plan, let $R > 0$ be the number of different paths each aircraft has access to. Given $1 \leq k_a, k_b \leq N$ two aircraft indexes and $1 \leq r_a, r_b \leq R$ two path indexes, let

- $C_{k_a, k_b, r_a, r_b} \in \{0, 1\}$ be the *conflict coefficient* between path r_a of aircraft k_a and path r_b of aircraft k_b . It is equal to 1 if these two paths are incompatible, and 0 otherwise.
- $V_{k_a, r_a} \in \{0, 1\}$ be the *validity coefficient* for path r_a of aircraft k_a . It is equal to 1 if and only if this path can be used. A path may be unusable because it is ill-defined or conflicting with an obstacle.
- $T_{k_a, r_a} \in \mathbb{R}^+$ be the travel time associated to path r_a of aircraft k_a .

The associated ILP problem is

$$\begin{aligned}
 & \underset{(x_{k,r})_{\substack{1 \leq k \leq N \\ 1 \leq r \leq R}} \in \{0, 1\}^{N \times R}}{\text{minimize}} && \sum_{\substack{1 \leq k \leq N \\ 1 \leq r \leq R}} T_{k,r} x_{k,r} && \text{(Cumulated travel time)} \\
 & \text{subject to} && \\
 & \forall k \in [1, N] \forall r \in [1, R] && x_{k,r} \cdot (V_{k,r} - 1) = 0 && \text{(Valid path),} \\
 & \forall k_a, k_b \forall r_a, r_b && C_{k_a, k_b, r_a, r_b} \cdot (x_{k_a, r_a} + x_{k_b, r_b}) \leq 1 && \text{(No conflicts),} \\
 & \forall k \in [1, N] && \sum_{r=1}^R x_{k,r} = 1 && \text{Unique solution per aircraft}
 \end{aligned} \tag{6}$$

The solver algorithm entirely relies on building this ILP problem and forwards it to a dedicated solver, and then extracts a path allocation from the solution. We assume we have access to a function called FitDubins such that $\text{FitDubins}(\vec{S}, \vec{E}, v, \rho, \tau)$ returns a list of straight extended paths from $\vec{S}$ to $\vec{E}$ with constant speed v , turn radius ρ and duration τ , based on the description in section 3.1.

We handle the dynamic obstacles set $\mathbb{O}$ introduced in problem 2 by assuming it can be split in two parts. The first is $\mathbb{O}_{\text{GEO}}$, the set *geometric* obstacles, which does not change with time and is described through its borders with straight lines and circles. The second part is the collection of dynamic obstacles, which are the other aircraft following Dubins paths, denoted $(P_j)_{1 \leq j \leq p}$. This characterization allows us to use the same functions to detect collisions with obstacles as we do for collisions between trajectories.

Algorithm 1: Dubins Fleet Planner algorithm

Data: Initial and final states $(\vec{S}_k, \vec{E}_k)_{1 \leq k \leq N}$; flight parameters $(v_k)_{1 \leq k \leq N}, (\rho_k)_{1 \leq k \leq N}, \delta$; time slots $(\tau_{k,m})_{\substack{1 \leq k \leq N \\ 1 \leq m \leq M}}$; static obstacles $\mathbb{O}_{GEO}$; other aircraft $(P_j)_{1 \leq j \leq p}$

Result: Dubins trajectories $(\gamma_k)_{1 \leq k \leq N}$

```

for  $1 \leq k \leq N$  do
     $L_k \leftarrow []$ 
    for  $1 \leq m \leq M$  do // Time slots
         $L_{k,m} \leftarrow \text{FitDubins}(\vec{S}_k, \vec{E}_k, v_k, \rho_k, \tau_{k,m})$ 
        Append  $L_{k,m}$  to  $L_k$ 
    foreach  $\gamma \in L_k$  do
         $v_{k,\gamma} \leftarrow 0$  if  $\gamma$  intersects with  $\mathbb{O}_{GEO}$  or is in conflict with one of the  $(P_j)_{1 \leq j \leq p}$ , and 1 otherwise
         $t_{k,\gamma} \leftarrow \text{len } \gamma / V$ 
if all  $(L_k)_{1 \leq k \leq N}$  are not empty then
    foreach different  $k_a, k_b \in [1, N]$  do in parallel
        foreach pair  $(\gamma_a, \gamma_b) \in L_{k_a} \times L_{k_b}$  do
             $c_{k_a, k_b, \gamma_a, \gamma_b} \leftarrow 0$  if  $\gamma_a$  and  $\gamma_b$  are separated, and 1 otherwise.
    Solve the ILP problem 6
    return the paths selected by the solver
return Failure: some aircraft have no paths.

```

Algorithm 1 described the *Dubins Fleet Planner* algorithm adapted to the ATC context. We chose the HiGHS solver by Huangfu and Hall [14] for solving the ILP problem.

A theoretical analysis of our algorithm shows several shortcomings. There are no guarantees that a valid conflict-free solution can be built from the chosen set of paths. The worst case complexity is dominated by solving the ILP problem, which is a well-known NP-Complete task. Complexity is exponential in the number of variables. The number of path per aircraft is $M \times |\Gamma|$, yielding a total of $N \times M \times |\Gamma|$ variables given to the ILP solver. To obtain a more reasonable figure, we compute complexity based on the number of path combinations for the fleet. We get $(M \times |\Gamma|)^N$, which is exponential in the number of aircraft N , but polynomial with respect to the number of time slots M and possible paths $|\Gamma|$.

We mitigate the algorithm limitations by first using a multithreaded implementation in C++. Furthermore, we limit the number of aircraft N to plan for simultaneously by restricting which aircraft can be replanned at a given time. These aspects are discussed in detail in the next section, which describes how this solver is used in an air traffic simulation around an airport.

4. Traffic management

Managing traffic arriving at and departing from an airport requires more than just planning it at a single point in time. We develop an automated traffic management system able to handle aircraft about to enter the TMA and incorporate them into the already planned aircraft without creating conflicts. We make several assumptions to simplify this problem. We suppose aircraft are not affected by wind or other perturbations, and follow perfectly the trajectory given in the ground referential. This implies precise trajectory tracking, which is realistic given that the trajectories we plan are similar to those used in actual operations. Furthermore, we require that each path begins and ends with a straight segment of set length. The start segment would allow continuous tracking despite any

delays that may be introduced by computation and communication times. The end one represents a part of the final straight of the landing procedure.

An aircraft may be rescheduled if it meets all of the following criteria:

- It is in the TMA;
- It is not nearing the end of its trajectory;
- It is going straight and isn't about to start a turn.

These criteria mainly limit the number of aircraft to be considered simultaneously for replanning, thereby keeping the problem complexity at acceptable levels for dynamic scheduling. The last one also ensures that no change of trajectory may happen during a turn.

Aircraft that are not rescheduled are taken into account by providing their flight paths as obstacles for the solver. If planning fails, it is retried at the next time step. Aircraft which already have a plan keep following it, and those which do not go straight along their initial direction. Aircraft which are yet to appear in the simulation are taken into account by supposing they will follow a straight flight plan.

Regarding time slots, we generate them by adding fixed offsets to the expected landing time. Offsets may be positive or negative, allowing respectively late and early landings. We also add time slots associated to the shortest possible paths. When an aircraft trajectory is updated, its expected time of arrival is also updated. This allows for further adjustments during subsequent replanning.

5. Case study

We validate our method by using real traffic data taken from the OpenSky historical database [15]. We chose to study the London Heathrow Airport. This airport has significant traffic with only two parallel runways. In usual operations, one runway is used for landings and the other for departures. Thus, there are no parallel landing operations, which are difficult to handle for our solver since the minimal horizontal separation is fixed. Furthermore, the Heathrow Approach Control uses four holding points, visible on fig. 4. The Dubins Fleet Planner does not rely on holding patterns, and this is of interest to investigate if it is able to find different solutions for managing the incoming traffic trajectories.

We used a full day of data, ranging from 4:00 a.m. on October 29, 2025, to midnight. We captured all the traffic in a 160 NM $\times$ 160 NM square centered on the Heathrow Airport, and kept only trajectories landing at or departing from this airport. Traffic data processing is achieved using the traffic Python toolbox [16].

5.1 Pre-analysis

We analyze the dataset to identify realistic values for the different parameters of our solver. We focus on three variables: minimal separation, speed, and altitude.

The International Civil Aviation Organization (ICAO) specifies in [17], section 8.7.3.2, that the minimal horizontal separation can be set to 3 NM when navigation systems are sufficiently accurate. And this separation can be lowered to 2.5 NM for aircraft aligned on the same final approach track. These two minima are in use at the Heathrow Airport, and analyzing our data revealed that the horizontal separation could be as low as 2 NM. Initial tests of our simulator with a minimal separation of 3 NM were unsuccessful. We ultimately decided to use a minimum separation of 2.7 NM, a compromise between the official recommendations and actual traffic conditions.

It is difficult to choose a constant speed, since airplanes slow down as they approach for landing and speed up as they take off. The speed distribution in our dataset is illustrated in fig. 3. The 2D histogram shows several behaviors for landing aircraft. In the last 25 NM, aircraft follow a set speed profile, corresponding to the landing procedures. But before this point, speed values range between 200 kt and 500 kt. There are two principal clusters, centered around 230 kt and 280 kt. For the sake of simplicity and uniformity, we picked the average 250 kt as the constant speed for landing aircraft in our simulation. Furthermore, we offset the trajectory endpoints by 3 NM before the runway threshold and require a final straight of 10 NM. These specifications represent a compromise between taking the final phase of the flight into account and simulating a flight path with inaccurate speed and separation requirements.

Regarding departing aircraft, the speed profiles are very similar from one aircraft to another for most of their flight paths. The main difficulty lies in choosing a single speed for the entire path. We choose 400 kt, as the histogram shows a peak around this value. For both departing and arriving aircraft, we have configured their turning radius based on the standard rate turn, so that the aircraft completes a full circle in 2 minutes at its assigned speed.

We analyzed altitude in the same way we analyzed speed, and observed similar behaviors. Departing aircraft all follow a similar climb profile, reaching 20 000 ft after traveling 50 NM. Incoming aircraft have varying descent profiles. There are several cruising altitudes between 10 000 ft and 16 000 ft. Over the last 40 NM before landing, the altitude profiles converge toward the final descent. Based on these observations, we have determined that we only need to plan the flight paths of departing aircraft up to a distance of 50 NM from the airport. Once this radius is reached, we consider departing aircraft to be vertically separated from arriving traffic.

5.2 From traffic data to simulation test case

To run our simulation, we need to convert the actual trajectories into test cases. We heavily relied on the traffic toolbox capabilities.

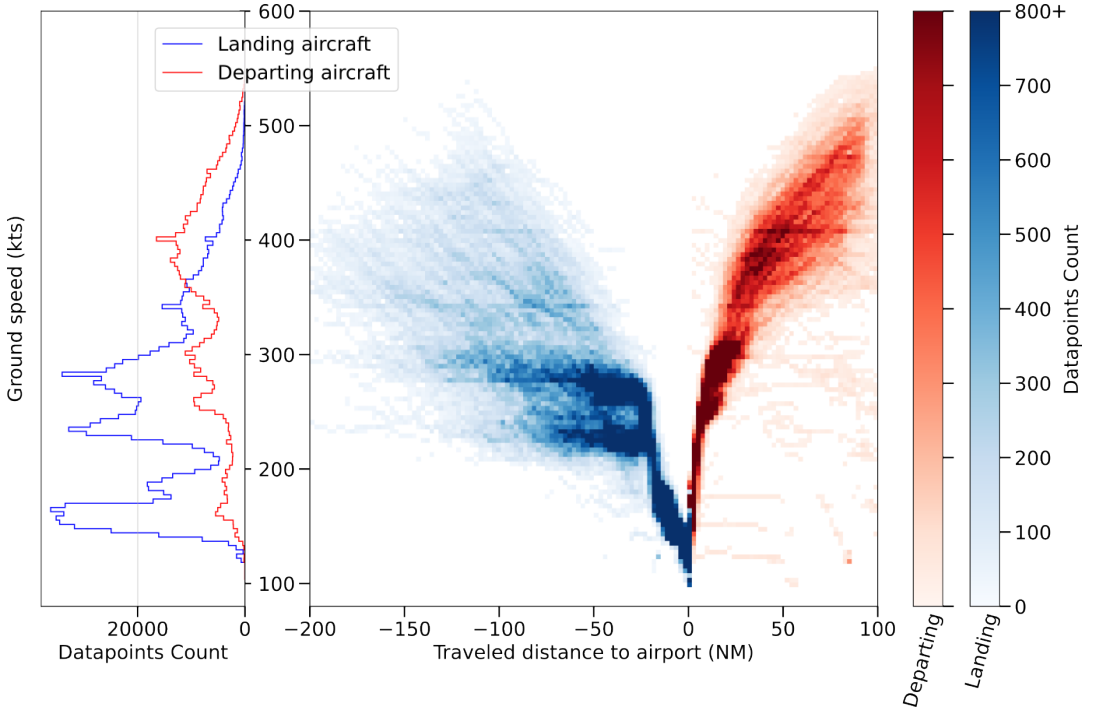
Out of the 1303 flights contained in our dataset, we kept 1280 for the simulation. 15 flights were removed because they were empty, and 8 because they were scheduled to land, but the runway could not be identified. Of the 1280 flights used in the simulation, 638 arrived at the airport and 642 departed from it.

For incoming flights, we set the initial position and orientation at the first known location, and the destination at 3 NM before the threshold of the destination runway, aligned with the runway centerline. For departing flights, the initial position is determined based on the first point detected during flight, and then shifted 3 NM forward to ensure there is no conflict with traffic landing on the other runway. The destination is defined as the last recorded position within a 50 NM radius, centered on the airport. In all cases, the timestamp associated to the chosen reference point is kept unchanged.

We define our TMA to be an 80 NM radius circle centered on Heathrow Airport. We configure the traffic management described in section 4 with the following parameters: trajectories must start with a straight segment 4 NM long and end with a straight segment 10 NM long. If an aircraft is less than 5 minutes away from a turn or its destination, it cannot be rescheduled. The time slots range from 30 minutes before to 30 minutes after the current arrival time, with a 3-minute interval between two consecutive slots. The simulation time step is set to 10 seconds.

5.3 Re-planning results

We ran our simulation on a computer equipped with 975 GB of RAM and an Intel® Xeon® Gold 6230 CPU @ 2.10GHz, using 80 threads (2 per core). The simulation took 737 seconds to run, with 2254



The traveled distance to airport is a signed value. When negative, it indicates the distance remaining before landing; when it is positive, it corresponds to the distance traveled since takeoff. The points located in the lower right portion of the 2D histogram correspond to incorrectly classified landing trajectories.

Figure 3. 1D (left) and 2D (center) histograms of the ground speeds measured in the dataset.

calls to the underlying solver implementing the algorithm 1. The resulting trajectories are shown in fig. 4. The runways at Heathrow Airport are in the center, and are used from east to west.

The flight paths of departing aircraft are mostly similar between actual and simulated traffic. They differ primarily in their respective starting points; the starting points of the simulated flight paths were shifted 3 NM away from the edge of the runway. For arriving traffic, there are two main differences. The four holding patterns used in real operation disappear in the simulation, and are replaced by deviations and spontaneous wait circles. Such circles are especially visible for trajectories coming from the north-west in the simulation. The other important difference concerns the traffic distribution. It is mostly visible on the eastern part, as our solver aligns the aircraft with the runways up to 80 NM away from the threshold. Another significant difference can be seen in the southwest section, where two incoming traffic streams are merged in the real data, but are not when using the solver. These differences explain how the Dubins Fleet Planner finds solutions without using fixed holding patterns. Instead, the DFP relies on small waiting circles at the trajectories starts, and large linear deviations. These linear deviations both delay and separate aircraft horizontally. This produces more variety in the airspace use, at the cost of occupying more area.

We use two main metrics: the first is to count the number of aircraft that reach their destination while maintaining the required minimum separation, and the second is to measure delays relative to the reference data. An overview of both is given in fig. 5. Out of the 1280 flights considered by solver, only 3 have been removed due to loss of separation. This loss of separation was caused by the DFP inability to find a conflict-free solution. These three events are spread along the day, happening

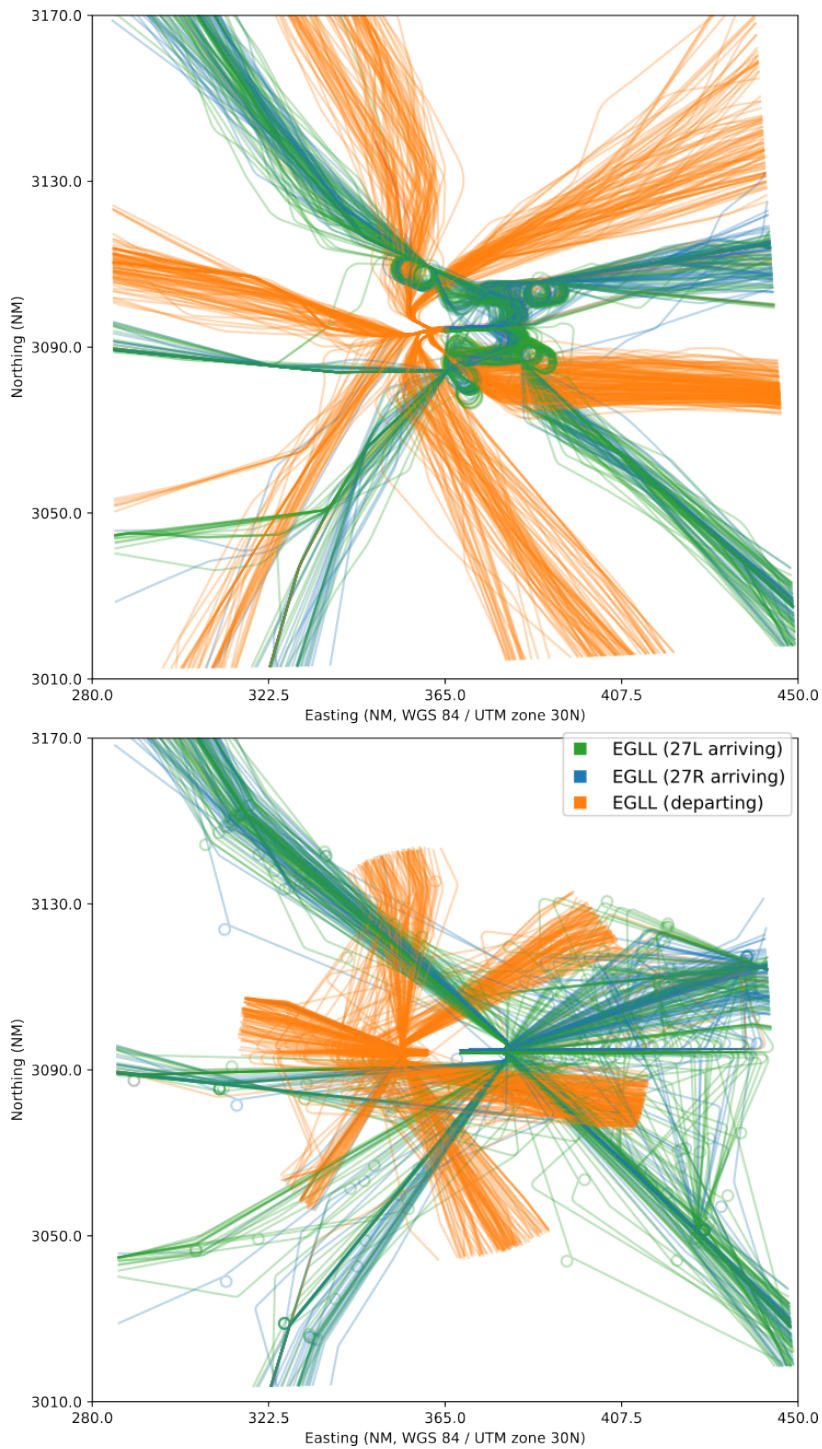


Figure 4. Illustration of traffic trajectories for the October 29th, 2025, around London Heathrow Airport (on top: measured traffic; below: replanned simulation).

around 8h30, 13h00 and 18h00 respectively.

Regarding delays, this chart shows three trends. Before 7h00, our simulated aircraft are late compared to their real counterparts. This is likely due to our speed setting being lower than the actual speed, as there is no congestion yet. Then, from 7h00 to 11h00, the DFP outperforms the real traffic for most aircraft. Because the DFP is not using the holding patterns, it may initially find better solutions. And finally, after 11h00, the traffic is either slightly in advance or late compared to the reference. This section is probably the most representative of the DFP performance.

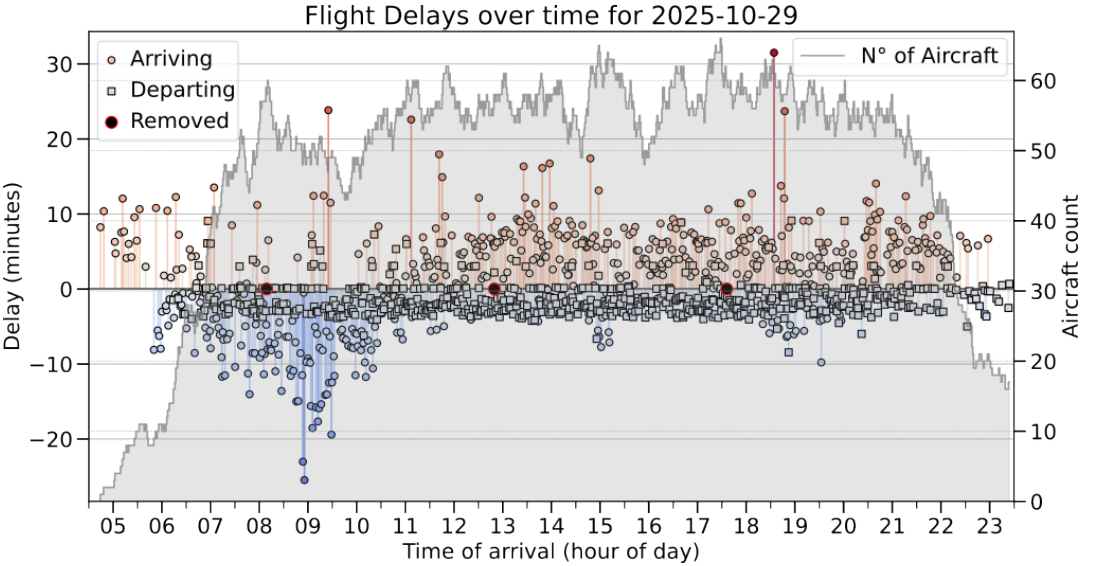


Figure 5. Air traffic trends throughout the day, showing the number of aircraft in flight over the past hour (shaded curve, right axis) and the difference in arrival times between the simulation and the reference (scattered points, left axis; above 0 means late, and below early). Three landing aircraft had to be removed from simulation due to loss of separation. They are represented by a black dot circled red.

We aggregate the delay data throughout the day to produce a summary, showcased in fig. 6. 75% of the departing aircraft reached their destination early, and 75% of arriving aircraft landed with less than 6 minutes of delay. On average, departing aircraft are 80 seconds early while landing aircraft are 80 seconds late. The variation in results is greater for aircraft during landing, with outliers at both extremes. Two planes landed more than 22 minutes early, and four others landed more than 22 minutes late.

6. Conclusions and perspectives

We presented the Dubins Fleet Planner, a flight path planning method based on generating different geometric flight paths for each aircraft and then selecting an optimal combination of conflict-free flight paths using an Integer Linear Programming formulation. Path generation has been adapted to match standard air traffic control trajectories. The trajectories consist of straight lines and standard rate turns, making them humanly readable.

The Dubins Fleet Planner has been tested on actual traffic data gathered from the OpenSky network. We tested this planner on a full day of traffic at the London Heathrow Airport. The solver was given 1280 flights to handle. It successfully planned most flights, failing to maintain a 2.7 NM horizontal separation for only 3 flights. On average, departing flights exited the Terminal Maneuvering Area

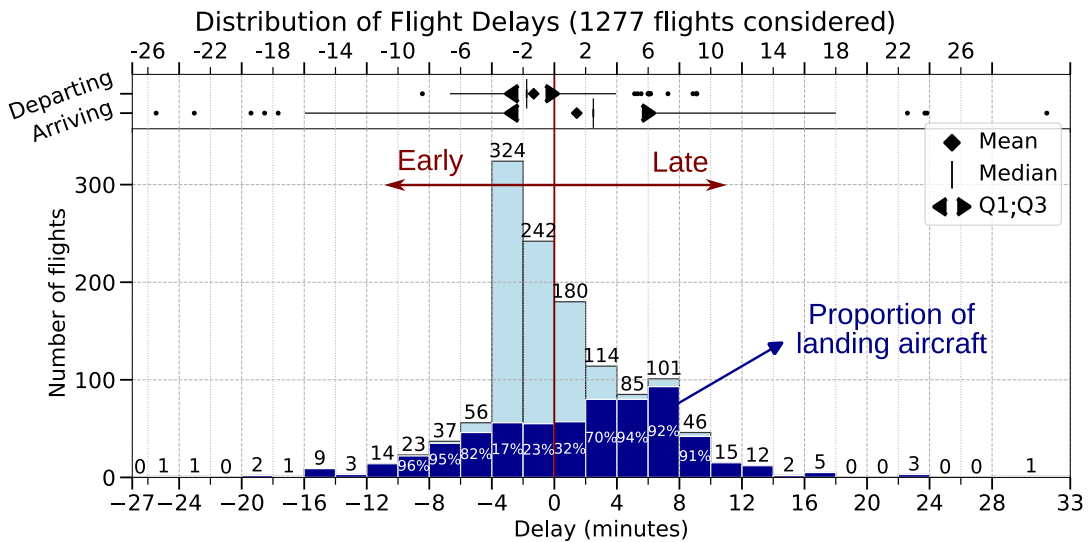


Figure 6. Distribution of differences in arrival times between simulated traffic and actual traffic. A positive delay indicates that the simulated flight arrives later than the actual arrival time, and a negative delay indicates that it arrives earlier. The top graph shows the distributions conditioned by the type of flight (departure or arrival), presented as box-and-whisker plots. The bottom graph shows the entire distribution as a histogram, with solid bars indicating the proportion of flights arriving at the airport.

80 seconds early, while arriving flights landed 80 seconds late. 75% of the flights managed by the schedulers reached their destination with less than 6 minutes of delay.

The Dubins Fleet Planner has several limitations. The main one is that there is no guarantee that a solution will be found. The others stem from the gap between assumptions and reality. Considering a variable speed would improve accuracy, especially for departing flights. Indeed, it is likely that these flights arrive early because their speed is overestimated. Similarly, delays for arriving flights could be reduced by flying at a higher speed at the beginning of the approach. The use of airspace can also be discussed. We did not take into account traffic that might fly through the Terminal Maneuvering Area of the Heathrow Airport terminal without landing there. Nor did we take into account any airspace restrictions, such as those due to populated areas or neighboring airports. Such restrictions can be implemented by using well-defined static obstacles.

There are several promising lines of improvement. The most critical aspect is to avoid loss-of-separation events. This problem could be mitigated by providing the solver with more possible paths, which would increase the chances of finding a solution. Another solution would be to reintroduce holding patterns and direct aircraft that have no other option to those patterns. With regard to speed, it is possible to relax the constraint of constant speed and switch to constant acceleration per segment without significantly compromising performance. Nevertheless, further research is needed to develop an effective mechanism for selecting the appropriate speed profiles. Finally, it should be possible to define arrival points that are sufficiently spaced apart to allow for parallel landings, thereby enabling us to test the Dubins Fleet Planner in more airports.

References

- [1] Xavier Olive, Junzi Sun, Luis Basora, and Enrico Spinielli. “Environmental inefficiencies for arrival flights at European airports”. In: *PLOS ONE* 18.6 (June 2023). Ed. by Simone Lolli, e0287612. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0287612.
- [2] EuroControl. *Point Merge – Improving and harmonising arrival operations*. 2006. URL: <https://www.eurocontrol.int/point-merge>.
- [3] Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive, and Emmanuel Rachelson. “The aircraft runway scheduling problem: A survey”. In: *Computers & Operations Research* 132 (Aug. 2021), p. 105336. ISSN: 0305-0548. DOI: 10.1016/j.cor.2021.105336.
- [4] Man Liang, Daniel Delahaye, and Pierre Marechal. “Conflict-free arrival and departure trajectory planning for parallel runway with advanced point-merge system”. In: *Transportation Research Part C: Emerging Technologies* 95 (Oct. 2018), pp. 207–227. ISSN: 0968-090X. DOI: 10.1016/j.trc.2018.07.006.
- [5] Dongdong Gui, Meilong Le, Zhouchun Huang, Junfeng Zhang, and Andrea D’Ariano. “Optimal aircraft arrival scheduling with continuous descent operations in busy terminal maneuvering areas”. In: *Journal of Air Transport Management* 107 (Mar. 2023), p. 102344. ISSN: 0969-6997. DOI: 10.1016/j.jairtraman.2022.102344.
- [6] Maël Feurgard, Gautier Hattenberger, Nicolas Durand, and Simon Lacroix. “Safe synchronous path planning for a fleet of fixed-wing constant speed aircraft”. working paper or preprint. Feb. 2026. URL: <https://hal.science/hal-05299362>.
- [7] Lester E. Dubins. “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents”. In: *American Journal of Mathematics* 79.3 (July 1957), p. 497. ISSN: 0002-9327. DOI: 10.2307/2372560. URL: <http://www.jstor.org/stable/2372560> (visited on 07/02/2025).
- [8] Madhavan Shanmugavel, Antonios Tsourdos, Rafał Żbikowski, and Brian White. “Path Planning of Multiple UAVs Using Dubins Sets”. In: (2005). DOI: 10.2514/6.2005-5827.
- [9] Aditya K. Rao, Kushal P. Singh, and Twinkle Tripathy. “Curvature bounded trajectories of desired lengths for a Dubins vehicle”. In: *Automatica* 167 (Sept. 2024), p. 111749. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2024.111749.
- [10] Zheng Chen, Kun Wang, and Heng Shi. “Elongation of curvature-bounded path”. In: *Automatica* 151 (May 2023), p. 110936. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2023.110936.
- [11] Weiran Yao, Naiming Qi, Jun Zhao, and Neng Wan. “Bounded curvature path planning with expected length for Dubins vehicle entering target manifold”. In: *Robotics and Autonomous Systems* 97 (Nov. 2017), pp. 217–229. ISSN: 0921-8890. DOI: 10.1016/j.robot.2017.09.003. URL: <https://www.sciencedirect.com/science/article/pii/S092188901730132X>.
- [12] Richard Peirce Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.
- [13] E. R. Hansen. “Global optimization using interval analysis: The one-dimensional case”. In: *Journal of Optimization Theory and Applications* 29.3 (1979), pp. 331–344. ISSN: 1573-2878. DOI: 10.1007/BF00933139.
- [14] Q. Huangfu and J. A. J. Hall. “Parallelizing the dual revised simplex method”. In: *Mathematical Programming Computation* 10.1 (2018), pp. 119–142. ISSN: 1867-2957. DOI: 10.1007/s12532-017-0130-5.
- [15] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. “Bringing up OpenSky: A large-scale ADS-B sensor network for research”. In: *IPSN-14 proceedings of the 13th international symposium on information processing in sensor networks*. IEEE, 2014, pp. 83–94.
- [16] Xavier Olive. “traffic, a toolbox for processing and analysing air traffic data”. In: *Journal of Open Source Software* 4 (2019), p. 1518. ISSN: 2475-9066. DOI: 10.21105/joss.01518.

- [17] ICAO. *Procedures for air navigation and air traffic management pans-atm doc 4444*. en. ICAO, 2016. ISBN: 9789292580810. 384
385

Author contributions 386

- Maël Feurgard: Methodology (lead), Software, Formal analysis, Investigation, Resources, Visualization, Writing. 387
388
- Nicolas Durand: Conceptualization (equally shared), Methodology, Writing (review and edition), Supervision 389
390
- Gautier Hattenberger: Conceptualization, Methodology, Writing (review and edition), Supervision 391

Open data statement 392

The software developed for this article is stored in two Github repositories: 393

- <https://github.com/spacewolfXfr/DubinsFleetPlanner.git> : Contains the C++ implementation matching algorithm 1. Depends on external libraries, most notably Boost for general algorithms and HiGHS for the ILP solver. 394
395
396
- https://github.com/spacewolfXfr/DFP_Ui : Collection of Python scripts for fetching data, using the Dubins Fleet Planner and visualizing results. It is based on the traffic Python environment. 397
398
The test dataset has been included in this repository at 399

`data/EGLL_20251029T040000_20251030T000000.parquet` 400

It can also be recovered by using the TrafficFetcher scripts, which sends a request to the OpenSky historical database. The parameters used are: 401
402

- Airport: EGLL 403
- Start: 2025-10-29 04:00 404
- End: 2025-10-29 23:59 405
- Maximal distance: 80 NM 406