6CCS3PRJ Final Year

Optimising cruise flight path trajectories to avoid the formation of contrail cirrus using Ant Colony Optimisation

Final Project Report

Author: Joseph Grabski

Supervisor: Dr. Josh Murphy

Student ID: 21078643

April 9, 2024

Abstract

Persistent contrail cirrus formed from the exhaust of an airplane contributes up to 35% of the aviation industry's total global emission impact. Diverting just 1.7% of these flights could potentially reduce the net-warming impact by 59.3%. In this report, a multi-objective Ant Colony Optimisation approach is explored that minimises the formation of contrail cirrus, total fuel consumption and total flight duration, for the cruise phase of trans-Atlantic flights. Average reductions of 0.79% for the flight duration, 2.21% for the CO_2 emissions, and 5416.97% for the total contrail EF were observed. The CoCiP prediction model is used to evaluate the formation of contrails, aircraft performance data and fuel flow models are provided by the *OpenAP* library, and weather data is retrieved from the ECMWF ERA5 data set.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Joseph Grabski

April 9, 2024

Acknowledgements

I would like to thank to my project supervisor, Dr. Josh Murphy, for his guidance and support throughout the project, as well as the *pycontrails* team for granting me access to their API for use in this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In order to tackle climate change, a concerted global effort is required, with the 2015 Paris Agreement [42] putting in place a goal to limit the average global temperature increase to $1.5°C$. To reach this goal, all sectors will need to find ways to reduce their total emissions and climate impact, including the aviation industry.

Despite only making up 2% of global CO_2 emissions, aviation is one of the most difficult industries to de-carbonise due to soaring post-COVID demand [18]. Therefore, technological innovation is necessary to find methods for reducing the industry's climate impact.

One of the largest contributors to aviation's climate impact is net-warming contrail formation, accounting for roughly *35%* [19] of the aviation industry's total climate impact, with contrail cirrus having a net warming effect about $1.5\times$ that produced from the aviation industry's CO_2 output [10]. It is also estimated that the global annual mean contrail cirrus net radiative forcing (RF) from 2018 alone, could exceed that of the forcing from aviation's cumulative CO_2 emissions since its inception (1940) [24].

As described by Appleman [3] contrails are formed when the warm engine exhaust and cold ambient air reaches saturation with respect to water, forming liquid drops, which quickly freeze. However, contrails only persist and cause a net warming effect when the relative humidity with respect to ice of the ambient air is greater than 100% [11] and less than 120% [39] within ice super-saturated regions (ISSR) in the troposphere. Diverting just 1.7% of flights could potentially reduce contrail energy forcing (EF) by 59.3%, with only a 0.014% increase in total fuel consumption and subsequent CO_2 emissions [40].

However, it is a uniquely difficult issue to optimise flight paths to avoid the formation of contrail cirrus, since avoiding ISSRs may cause a flight to be delayed (adding extra stress to

flight scheduling), or increase fuel consumption, negating the positive effects offered by less contrail energy forcing.

Therefore, due to the climate emergency, it is of growing interest to develop algorithms that optimise flight paths using a multi-objective function to avoid ISSRs, reduce fuel costs, and to minimise total flight time - with the hope of reducing the industry's net warming effect.

The aim of this project is to explore a novel approach to avoid ISSRs during the cruise phase of a flight, using a multi-objective Ant Colony Optimisation algorithm (m-ACO) [17], whilst also minimising total fuel consumption and total flight time. The approach taken will be primarily extend the method described by Alejandro Murrieta-Mendoza et al. [26], which optimises both the lateral and vertical trajectory during the cruise phase of the flight to reduce fuel consumption.

Contrail formation will be evaluated based off weather data provided by the ECMWF ERA5 data-set [7], and the CoCiP prediction model [32]. Aircraft performance data and fuel flow models are provided by the *OpenAP* [38] library.

Ultimately, the results of the optimisation process will then be compared against random flight paths, and real-world flight paths from *FlightAware* [13] to compare fuel consumption, flight time, and climate impact.

Chapter 2

Background

2.1 Contrail Formation and their Environmental Impact

Contrails form when water vapour from an aircraft's engine exhaust condense and freeze on soot particles from the engine's exhaust. In a dry atmosphere, the ice quickly forms creating a short cirrus (cloud made of ice crystals) that dissipates quickly and is only a few kilometres long, resulting in a negligible environmental impact. For a fraction of flights though, they pass through regions that are supersaturated with respect to ice (ISSRs), where ice crystals also form from the water vapour in the surrounding atmosphere. This results in the formation of persistent contrails that can extend for hundreds of kilometres with a lifetime of several hours. A portion of these persistent contrails absorb long-wave radiation, and reflect short-wave solar radiation, resulting in a net-warming effect due to the greenhouse effect [10]. However, some contrail cirrus can have a net-cooling effect during the day, and dependant on weather conditions [48].

Typically, man-made climate impact is measured using radiative forcing (RF) [43]. However, a better measure used specifically to measure the impact from contrail cirrus is energy forcing (EF), defined as the time integral of local RF weighted by the width of the contrail [33]. Using a contrail prediction model [32], EF estimates can be calculated for a given flight path.

2.2 Geodesic Paths

A geodesic path is described as the shortest path between two points on a surface. Within the context of the globe, the path can be calculated by finding a series of waypoints along the great

circle path between two geographic points [47]. This provides a series of points on a sphere, which can then be converted to points on the WGS84 geodetic system [2] (a standard reference system, used for GPS). Alternatively, Vincenty's formulae [20] can be used to calculate the geodesic path without a need for a conversion step.

2.3 Flight Equations of Motion

Many characteristics of the aircraft during flight need to be calculated and predicted based on weather and thrust data, as well as the initial configuration of the aircraft. This includes: true air speed (TAS), ground speed (GS) and crabbing angle (α_{CRB}).

The true air speed can be calculated either using the MACH number, or with the indicated air speed (IAS). To calculate TAS based off the MACH number, it is first necessary to calculate the speed of sound (a_0) and the air temperature ratio (Θ):

$$a_0 = M\sqrt{\gamma RT} = M\sqrt{\gamma RT_0}\sqrt{\Theta}$$

$$\Theta = \frac{T}{T_0}$$

where M is the MACH number, γ is adiabatic index for air (1.4), R is the universal gas constant for dry air ($28705 J/kg/K$), T is the air temperature in Kelvin and T_0 is the standard air temperature at sea level ($288.15K$). From this, a_0 is found to be $340.29 m/s$. The TAS can then be found with:

$$TAS = Ma_0\sqrt{\Theta}$$

Alternatively, to find the TAS from the IAS, the dynamic pressure for the IAS at sea level in ISA conditions needs to be found.

$$q_c = p_{s0}\left\{\left[1 + 0.2\left(\frac{IAS}{a_0}\right)^2\right]^{3.5} - 1\right\}$$

where q_c is the dynamic pressure, and p_{s0} is the static air pressure at sea level ($101,325 Pa$). The TAS is then defined as:

$$TAS = a_0\sqrt{5\Theta\left[\left(\frac{q_c}{p_s} + 1\right)^{3.5} - 1\right]}$$

where p_s is the static air pressure at the flight altitude.

For calculations regarding distance travelled and time elapsed, the GS is used, which is the TAS corrected for wind. The GS is defined as:

$$GS = \left(W_V \cos\alpha_{Segm} + W_U \sin\alpha_{Segm}\right) + \sqrt{(TAS\cos\alpha_{CD})^2 - (W_V \sin\alpha_{Segm} - W_U \cos\alpha_{Segm})^2}$$

where W_V and W_U are the northward and eastward wind components respectively, α_{CD} is the aircraft's climb/descent angle, and α_{Segm} is the course or segment heading at the current location. This is the direction that the aircraft would travel in the absence of wind.

However, to follow this same trajectory in windy conditions, the wind triangle can be used to calculate the "crabbing angle", which is summed to the course to obtain the heading the aircraft should follow to maintain its intended direction.

$$\alpha_{CRB} = \frac{180}{\pi} \arcsin\left(\frac{W_V \sin\alpha_{Segm} - W_U \cos\alpha_{Segm}}{TAS}\right)$$

2.4 Ant Colony Optimisation

2.4.1 Ant Colonies in Nature

Ant Colony Optimisation is modelled off the behaviour of ants finding food in nature. Ants will wander an area, and upon finding food (F), they will return to the nest (N) with the food. As they travel, ants will leave behind a chemical pheromone trail. Ants from the same colony can detect this pheromone trail, and if the pheromone concentration is high enough, they will follow the pheromone trail to the same food source that the original ant found.

Overtime, the pheromone trail will evaporate, meaning that a trail needs to be frequently travelled for it to maintain its pheromone concentration. If a trail has too low a concentration, a decreasing number of ants will follow it until it completely disappears.

Not all ants will follow the exact path of the pheromones either, meaning they can potentially deviate and find a shorter path to the food source. In this case, they will be able to travel between the food source and the colony more than ants on the other trail in the same time-frame, gradually increasing the pheromone concentration to the point where more ants will choose the new trail over the initial one. Eventually, ants will converge on a global optimum to the food source, and the pheromone concentration will be strongest until the food has all been

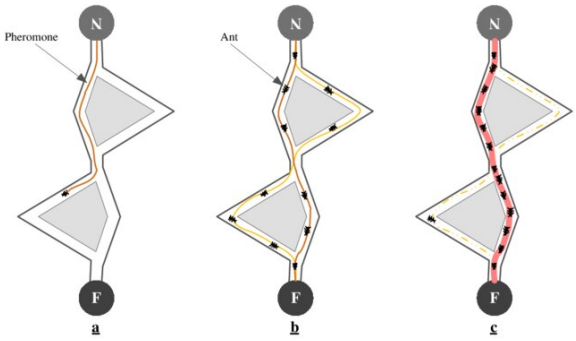


Figure 2.1: Ant colonies finding food in nature from Mendoza et al. [26]

transferred to the colony, as can be seen in Fig. 2.1

2.4.2 Multi-Objective Ant Colony Optimisation

The Multi-Objective Ant Colony Optimisation algorithm (m-ACO) [17] is an extension of the Ant Colony Optimisation (ACO) algorithm [36]. Specifically, the m-ACO variant used in this project uses one colony and multiple pheromone structures (m-ACO$_4(1, m)$). The algorithm aims to mimic the behaviour of ants in nature as described in the previous section, where ants leave pheromones on the edges of a graph $G = (V, E)$, whilst also optimising several defined objective functions.

The algorithm can be split into two parts, solution construction and updating pheromone values.

As seen in Fig. 2.2, a solution is first constructed for each ant by sequentially selecting a candidate vertex $Cand$ from the set of vertices V with probability $p_S(v_i)$. Any vertices that violate a constraint are removed, for example if they are outside of the search space. $p_S(v_i)$ is defined as follows:

$$p_S(v_i) = \frac{[\tau_S(v_i)]^\alpha \cdot [\eta_S(v_i)]^\beta}{\sum_{v_j \in Cand} [\tau_S(v_j)]^\alpha \cdot [\eta_S(v_j)]^\beta}$$

where $\tau_S(v_i)$ is the pheromone factor, and $\eta_S(v_i)$ is the sum of all heuristic factors at the candidate vertex v_i, such that $\eta_S(v_j) = \sum_{i=1}^{m} \eta_S^i(v_i)$. α and β are the weightings of these two factors, and can be adjusted to favour one factor over the other.

After solutions for all ants are constructed, pheromone concentrations are updated for the vertices of the best solution. ρ is the evaporation factor, such that $0 \leq \rho \leq 1$. The new quantity of pheromone at a component c is described as:

Figure 2.2: m-ACO Algorithm

```
procedure M-ACO(#τ)
    Initialize all pheromone trails to τ_max
    repeat
        for all ants do
            CONSTRUCT-SOLUTION
        end for
        for i to #τ do
            update i
            if i < τ_min then
                i ← τ_min
            end if
            if i > τ_max then
                i ← τ_max
            end if
        end for
    until maximal number of iterations reached
end procedure

function CONSTRUCT-SOLUTION
    S ← ∅
    Cand ← V
    while Cand ≠ ∅ do
        choose v_i ∈ Cand with probability p_S(v_i)
        add v_i at the end of S
        remove from Cand vertices that violate constraints
    end while
end function
```

$$\tau^i(c) \leftarrow (1-\rho) \cdot \tau^i(c) + \Delta\tau^i(c)$$

$\Delta\tau^i(c)$ is the amount of pheromone to be deposited on a component c, defined as:

$$\Delta\tau^i(c) = \begin{cases} \frac{1}{1+f_i(S^i)-f_i(S^i_{\text{best}})} & \text{if } c \text{ is a component of } S^i \\ 0 & \text{otherwise} \end{cases}$$

where f_i is the i^{th} objective function, S_i is the solution that minimises f_i and S_{best} is the solution that minimises all objective functions f since the beginning of the run.

2.5 Literature Review

There is a substantial body of research into the problem of optimising the trajectory of an aircraft. The approaches taken can be separated into a number of sub-categories:

- Abstract Obstacle-Avoidance

- Altitude Only Adjustment
- Traditional Path-Finding
- Multi-Objective Trajectory Optimisation
- Evolutionary Algorithms

A non-exhaustive selection of these will be explored in the following section.

2.5.1 Abstract Obstacle Avoidance

One potential approach for avoiding contrail formation, is to model ISSRs as restricted areas, or hostile threats. There is a significant amount of research into optimising the paths of drones to navigate a 3D environment to avoid threats. For example, Yaohong Qu et al. [30] constructed a 3D Delaunay diagram and then modeled a graph by connecting the midpoints of the convex polyhedron. Dijkstra is then ran on the graph to find the shortest path. To then consider factors such as pitch angle and steering angle, the Artificial Potential Field algorithm is then used to further optimise the path, using a series of repulsive and attractive forces.

Similarly, recent research from Xijian Bian et al. [46] optimised fuel costs in an abstract environment with obstacles and avoidance zones. For their approach, a Maklink diagram was constructed, and then navigated using Dijkstra to find the shortest path. A genetic ant colony algorithm is then used to further optimise this path according to a collision probability analysis.

However, the primary issue with both approaches applied to avoiding contrail formation, is that it is not always possible or optimal to completely navigate around ISSRs. The fuel penalty from laterally navigating around these regions can in fact exceed the potential environmental benefits [31]. There is also the potential that navigating through the ISSR produces a net-cooling effect too [48], which cannot be modelled by taking this approach.

2.5.2 Altitude Only Adjustment

To combat the issues surrounding the added fuel cost from navigating laterally around ISSRs, many approaches aim to optimise existing flight paths by only increasing the cruise altitude by either $2000ft$ or $4000ft$ increments [4]. For example, the approach taken by Denis Avila et al. adjusts the cruise level of contrail generating segments of flight paths from over the USA by $2000ft$, and then by another $2000ft$ (total $4000ft$) if the new cruise level still generates contrail cirrus. To calculate whether a segment generates a contrail cirrus, the Schmidt-Appleman

criterion [3] for ISSRs is used. Using this approach, there was a 92% decrease in the net RF produced with a 4000ft altitude increase. Moreover, fuel consumption did not increase, due to the lower drag at higher altitudes.

Likewise, Roger Teoh et al. selectively diverted just 1.7% of flights over Japan, to reduce the total contrail cirrus EF by 59.3%, with a 0.014% increase in total fuel consumption and CO_2 emissions. The approach taken to divert trajectories is similar, with multiple new segments being generated at altitudes of $\pm 2000ft$. The trajectory with the lowest EF is then selected as the diversion. To calculate the EF of a trajectory, the CoCiP prediction model [32] is used.

These approaches have been shown to be effective at reducing the formation of contrail cirrus, however the original reference flight paths are not optimal, and can also be improved laterally to save on fuel costs alongside the reduction of contrail cirrus. A multi-objective function cannot be modelled using these approaches, and therefore may favour an approach using an evolutionary algorithm.

2.5.3 Traditional Path-Finding

An interesting approach taken by Florent Vergnes et al. [44] applies the A* path-finding algorithm to the problem of flight path optimisation, for both fixed grids and free routes. The A* algorithm aims to minimise total flown distance, total burnt fuel, CO_2 and non-CO_2 emissions, whilst also assuming a constant MACH and flight level along the trajectory to reduce the potential search space. Another key approach to reduce the search space is the use of an "exploration cone" to determine which nodes should be expanded by the A* algorithm, since it is not necessary to expand nodes that are behind the current waypoint for example. The cost function is a past-dependent cost function, taking into account fuel cost from the departure node to the current and previous nodes. The heuristic is then the cost from the current node to the final node, assuming a direct route is taken and ignoring network constraints. Fuel and emissions are estimated using the BADA model [28] and *OpenAP* [38] respectively. Using this approach a 5% reduction in CO_2 and NO_x emissions was observed.

However, as mentioned within the paper, contrail cirrus is not included within their method which would significantly expand the potential search space. This is addressed as an area for future research, which is being explored as part of this report. The MACH number is also assumed to be constant, reducing the search space but leaving room for potentially more optimisation. It is also not possible for the aircraft to change flight level halfway through the flight plan, which as previously discussed is an important strategy that should be explored

for contrail cirrus reduction. The risk with adding this to the search space is a combinatorial explosion, which requires excessive pruning of the search space. The benefit of the approach taken by the authors though, is that optimal trajectories can be returned within a reasonable time ($\sim$ 1 minute).

2.5.4 Multi-Objective Trajectory Optimisation

Several linear-programming attempts to multi-objective trajectory optimisation (MOTO) were summarised and explored by Gardi et al. [14], and implemented by Lim et al. [48]. It was found that the use of MOTO applied to the avoidance of contrail formation zones and CO_2 reduction was effective, with a significant reduction in net RF and CO_2 emissions observed in the case-study. The use of MOTO too allowed for ISSRs to exploited during the day to result in a negative RF effect. This is a result of using the CoCiP [32] model, rather than simply avoiding segments using the Schmidt-Appleman criterion (SAC) [3], and a benefit of this approach over other contrail-avoidance algorithms that rely solely on SAC. [15]

Another important innovation is the use of a 4D field for contrail prediction [22]. Several linear trajectories were generated, and then interpolated between to create a 2D map of contrail formation at each altitude. Several 2D maps were then interpolated to create a 3D volume, and then finally several 3D volumes at different time-steps were interpolated to generate a 4D field. The 4D field can then be used as a look-up table when evaluating trajectories, presenting a significant optimisation, especially since the search space is known a priori.

2.5.5 Evolutionary Algorithms

The primary approaches using evolutionary algorithms are from Mendoza et al. [9, 25, 26, 27]. Genetic, bee-colony, ant-colony and particle swarm optimisation techniques were all explored and applied to optimising flight trajectories for total fuel consumption. The reduction of contrail cirrus was not considered.

Regarding the construction of the lateral flight plan, two different approaches are taken. Firstly a static grid approach [9, 26, 27] is used, where a grid of potential waypoints are constructed along the geodesic path between two points, which a flight path can then be selected from. When using this approach, it was found that rather than randomly selecting a consecutive node at each waypoint, it is better to use longer segment lengths with consistent lateral step values. This results in flight paths that are more consistent with regular flight navigation and resemble a zig-zag pattern less.

The dynamic grid approach [25] taken defines a search space, and allows the evolutionary algorithm to mutate on a continuous displacement between consecutive waypoints, rather than a discretised grid. Trajectories are generated either by placing a new trajectory parallel to the geodesic path but offset laterally, or by arbitrarily selecting waypoints from the geodesic path and randomly offsetting their positions. A maximum lateral offset is used to prevent impossible aircraft heading changes, requiring surrounding points to be corrected to fit this constraint. New waypoints are constrained to the initial bounds too to prevent waypoints drifting too far from the geodesic path.

A strict time constraint was placed [25, 26] on potential flights, meaning that any flights that are longer than the reference flight are immediately discarded. For the approach taken this is a necessary adjustment, due to a secondary MACH optimisation step that occurs after the initial flight optimisation. However, this constraint is too strict for a multi-objective problem where it may be advantageous for certain objectives to allow slightly longer flight paths, if other objectives are significantly improved.

Chapter 3

Requirements & Specification

3.1 Brief

The goal of the project is to create an implementation of the m-ACO algorithm that can optimise a flight route between any two points on the WGS-84 world model. The aim of this optimisation is to reduce fuel consumption, contrail cirrus formation, and the total flight duration. The model which the algorithm explores should also be easily decoupled, to potentially allow different evolutionary algorithms to be tested in the future.

3.2 Requirements

The requirements are divided into two sections: the requirements for the model, and the requirements for the evolutionary algorithm.

3.2.1 Model Requirements

Weather Model Requirements

- The weather data must be provided in a suitable format
- Weather data must be provided for several different pressure levels
- Weather data must be available at several different timestamps
- A suitable subset of the weather data must be used based on the chosen flight path
- A suitable resolution must be used to provide accurate weather data

- The weather data must be linearly interpolated when accessing points that are more precise than the chosen resolution

Contrail Formation Model Requirements

- A suitable prediction model must be used to simulate contrail formation
- Contrail formation must be stored as a 4D field containing information on EF levels, contrail lifetime and contrail size

Fuel Flow Model Requirements

- The fuel flow model must provide accurate fuel flow data at several different altitudes and thrust values for a variety of aircraft
- The fuel flow model must provide accurate flight envelope information for a variety of aircraft and engines

3.2.2 Optimisation Algorithm Requirements

Flight Path Construction Requirements

- Two points must be able to be selected to optimise a flight path between
- A suitable grid of waypoints must be constructed between two points on the globe
- The grid of waypoints must be 3D, and have points at several different altitude steps
- A graph must be constructed using the grid of points, with heading and altitude constraints determining which nodes have edges between them
- The constructed flight path must respect the aircraft's heading limits when selecting a consecutive point from where it currently is
- A selection of suitable aircraft flight characteristics must be calculated for each point along the flight path
- A sensible upper and lower limit must be placed on the altitude of the flight path

Ant Colony Optimisation Requirements

- The ACO algorithm must consider multiple objective functions

- The algorithm could calculate an optimal flight path within under an hour
- The algorithm should produce a flight path that is more optimal on at least one objective function than a reference trajectory
- The algorithm must produce a flight path that fits within the constraints of the aircraft's flight envelope.

3.3 Specification

3.3.1 Model Specification

Weather Model Specification

The weather model will be based off GRIB data provided by the ECMWF ERA5 dataset [7]. Weather data is obtained at three different pressure levels (300Pa, 250Pa and 200Pa), and at hourly timesteps across a 12 hour timespan. Weather data will be obtained for wind vectors, relative humidity with respect to water and air temperature. The grid resolution is 0.25 deg. ‘

Contrail Formation Model Specification

The CoCiP prediction model [32] will be used for contrail formation predictions and calculations. Although, a pre-calculated 4D contrail grid, inspired by Lim et al. [22], will be used during the optimisation process as a heuristic. After the optimisation process is complete, the CoCiP model will then be run as a more accurate prediction of contrail size, lifetime, and EF.

Fuel Flow Model Specification

OpenAP [38] will be used as the fuel flow model for this project, as it provides a free and open-source library with data on various engine and aircraft types, based off the Boeing Fuel Flow model. Primarily, it will be used to calculate the fuel flow of the aircraft at different stages of the flight, however it is also possible to calculate both CO_2 and NO_x emissions using this library, which will be used as part of an objective function.

3.3.2 Optimisation Algorithm Specification

Flight Path Construction Specification

To construct the flight path, a routing grid will be constructed along the geodesic path between two specified points. The approach to construct the 3D routing grid will be modelled off the

approach by Mendoza et al. [9, 26], with constraints meaning the aircraft can only ascend during the cruise phase.

A routing graph will be constructed using a consecutive points algorithm to determine which nodes should have edges between them based off heading and altitude constraints. Optimisation is then simpler, as these constraints no longer need to be considered as they have been baked into the routing graph.

Ant Colony Optimisation Specification

Several objective functions will be defined, based off emissions data from the APM, contrail cirrus EF from CoCiP, and time of arrival calculations based off the flight equations of motion. Multi-processing will be used to attempt to bring the total processing time under one hour. The resulting flight path will also be tested against the flight envelope from the APM to ensure it validates this constraint, otherwise it will be discarded.

Moreover, the flight path calculated from the m-ACO algorithm will be compared against a reference trajectory from *FlightAware* [13] on the same objective functions to ensure that it is more optimal.

Chapter 4

Design

Aspects of the design have already been covered as part of the Background section, however it is important to establish exactly what will be included as part of this project.

4.1 Model Construction

In order to construct potential solutions for the m-ACO algorithm, a model needs to be established. This consists of a static grid of waypoints to search through, weather data interpolation for the grid points, and finally the construction of a flight path.

4.1.1 Grid Construction

To construct a flight path, a static grid approach is used similar to that described by Mendoza et al. [9, 26]. A grid of waypoints are generated between a departure and destination point on the Earth which can then be used as a base for the m-ACO to search through and generate an optimal flight path.

Constructing a grid of waypoints occurs in four separate stages, eventually resulting in a graph which potential flight paths can be generated using. Each stage is described in the following section:

1. Geodesic Path Calculation
2. Routing Grid Construction
3. Altitude Grid Construction
4. Routing Graph Construction

Geodesic Path Calculation

The geodesic path between the destination and departure points is used as the baseline path, from which the rest of the grid is constructed. To calculate this path, the great-circle equations are used, and then the series of points generated are converted to the WGS84 co-ordinate reference system (CRS) using PROJ4 [29].

To find a waypoint along the great circle path, it is first necessary to calculate the course from the departure point $P_1 = (\phi_1, \lambda_1)$ to the destination point $P_2 = (\phi_2, \lambda_2)$ (where ϕ is the latitude and λ is the longitude, both in radians). The course at the departure point is subsequently defined as:

$$\tan \alpha_1 = \frac{\cos \phi_2 \sin \Delta\lambda}{\cos \phi_1 \sin \phi_2 - \sin \phi_1 \cos \phi_2 \cos \Delta\lambda}$$

where $\Delta\lambda = |\lambda_2 - \lambda_1|$

The central angle o_{12} between the two points is then found by:

$$\tan o_{12} = \frac{\sqrt{(\cos \phi_1) \sin \phi_2 - \sin \phi_1 \cos \phi_2 \cos \Delta\lambda)^2 + (\cos \phi_2 \sin \Delta\lambda)^2}}{\sin \phi_1 \sin \phi_2 + \cos \phi_1 \cos \phi_2 \cos \Delta\lambda}$$

We then extrapolate the great circle back to the point where it crosses the equator at node A. The azimuth at A, α_0 is given by:

$$\tan \alpha_0 = \frac{\sin \alpha_1 \cos \phi_1}{\sqrt{\cos^2 \alpha_1 + \sin^2 \alpha_1 \sin^2 \phi_1}}$$

The angular distance from A to P_1, o_{01} can then be found by:

$$\tan o_{01} = \begin{cases} \frac{\tan \phi_1}{\cos \alpha_1} & \\ 0 & \text{If } \phi_1 = 0 \text{ and } \alpha_1 = \frac{1}{2}\pi \end{cases}$$

The longitude at the equator, λ_{01}, is given by:

$$\tan \lambda_{01} = \frac{\sin \alpha_0 \sin o_{01}}{\cos o_{01}}$$

The position and azimuth at an arbitrary point P along the great-circle path between P_1 and P_2 can then be calculated with Napier's rules [35]:

$$\tan \phi = \frac{\cos \alpha_0 \sin o}{\sqrt{\cos^2 o + \sin^2 \alpha_0 \sin^2 o}}$$

$$\tan(\lambda - \lambda_0) = \frac{\sin \alpha_0 \sin o}{\cos o}$$

$$\tan \alpha = \frac{\tan \alpha_0}{\cos o}$$

where $o = o_{01} + \frac{d}{R}$, to find a point a d distance from the starting point and R is the radius of the Earth ($R \approx 63711km$).

This is repeated for an arbitrary number of steps along the great circle path to generate evenly-spaced waypoints. Each waypoint is then mapped from a sphere to the WGS84 CRS using PROJ4 [29].

Routing Grid Construction

To then construct the routing grid from the geodesic path, several offset waypoints are calculated either side of each waypoint along the geodesic path. Each of the offset points are arbitrarily $100km$ apart from each other. To calculate the position of the offset points, it is first necessary to calculate the bearing (Θ) from the current point to the next point with:

$$\theta = \text{atan2}(\sin \Delta\lambda \cdot \cos \phi_2, \cos \phi_1 \cdot \sin \phi_2 - \sin \phi_1 \cdot \cos \phi_2 \cdot \cos \Delta\lambda)$$

The angle perpendicular to Θ, is defined as $\Theta' = (\Theta + \frac{\pi}{2}) \mod 2\pi$. A point $100km$ perpendicular to the course of the current waypoint along the geodesic path is founding by using two equations for the latitude ϕ_n and the longitude λ_n:

$$\phi_n = \sin^{-1}(\sin \phi_1 \cdot \cos \delta + \cos \phi_1 \cdot \sin \delta \cdot \cos \theta)$$

$$\lambda_n = \lambda_1 + \text{atan2}(\sin \theta \cdot \sin \delta \cdot \cos \phi_1, \cos \delta - \sin \phi_1 \cdot \sin \phi_n)$$

This is repeated at every point, an arbitrary number of times either side of the geodesic path. The number of points either side are steadily increased and decreased as the algorithm moves away from the departure point and towards the destination respectively. This reduces the search space, since it would not be optimal for an aircraft to make sudden shifts in course, meaning these points would not be part of the final solution even if they were considered [26]. Figure 4.1 illustrates the construction of a routing grid between the London Heathrow Airport and John F. Kennedy Airport.

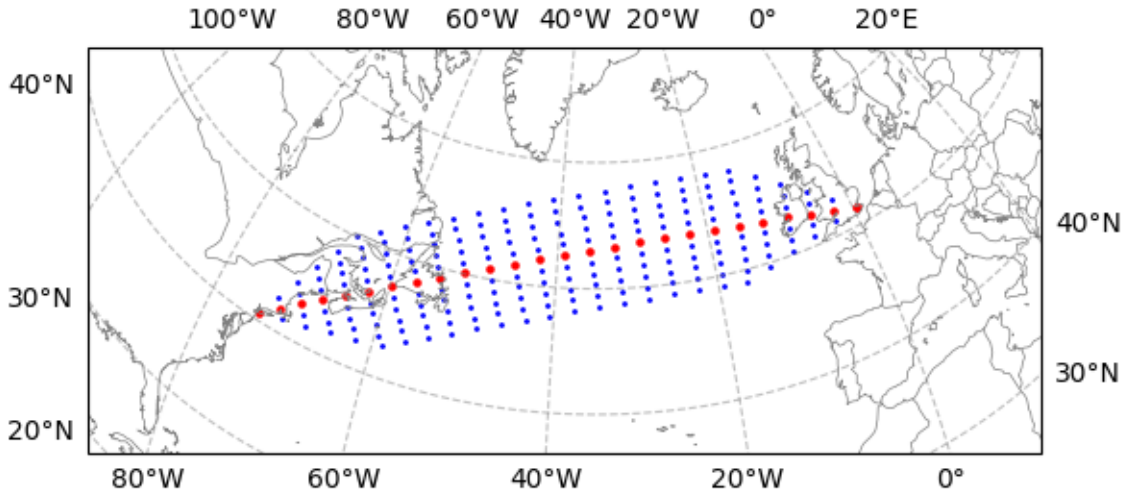


Figure 4.1: Routing grid constructed between LHR and JFK, with offset points in blue and the geodesic path in red

Altitude Grid Construction

The altitude grid is then constructed from the routing grid by placing points at regular $2000ft$ intervals above a base altitude, until a maximum altitude is reached. Since the aircraft can only ascend $2000ft$ between two waypoints, points from the departure point are discarded if it would require an increase in altitude greater than $2000ft$. Also, a point is considered a valid destination as long as it matches the final latitude and longitude, regardless of what altitude it is at.

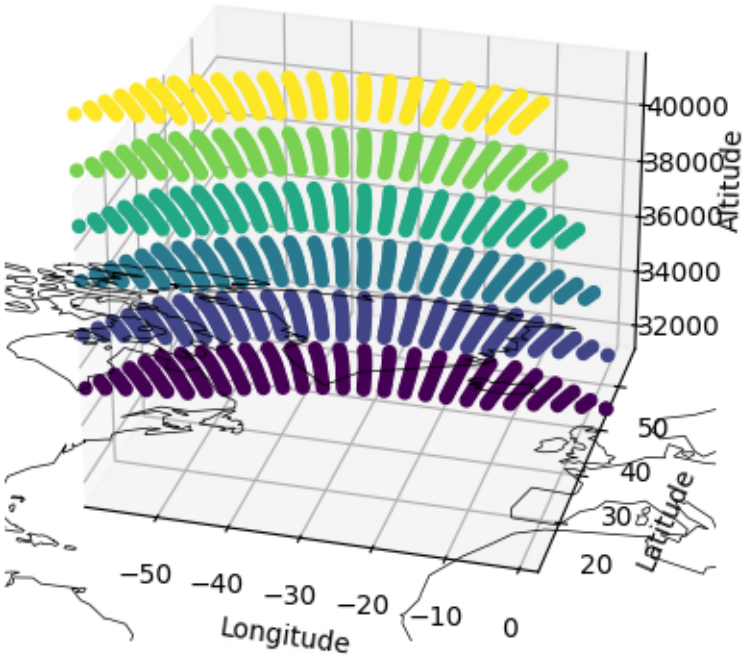


Figure 4.2: Altitude grid constructed from the previous routing grid in Figure 4.1.

Routing Graph Construction

To make the altitude grid easier to traverse, a unidirectional directed *NetworkX* [16] graph is constructed using a consecutive points algorithm specified in Figure 4.3 to create edges between this node and neighbours it should be able to access. The resulting graph has pheromone values initialised at every edge, and heuristic values at every node. Heuristic values are calculated at each node, using every objective function.

Figure 4.3: The consecutive points algorithm used to find neighbouring points

procedure GET_CONSECUTIVE_POINTS($x_i, y_i, altitude, grid$)
  $altitude_{max} \leftarrow altitude + MAX\_VAR$
  $altitude_{max} \leftarrow \max(STARTING\_ALTITUDE, \min(altitude_{max}, MAX\_ALTITUDE))$
  $points \leftarrow \emptyset$
  $altitude_i \leftarrow altitude$
  while $altitude_i \leq altitude_{max}$ **do**
    if $x_i + 1 == \text{len}(grid[altitude_i])$ **then**
      return $\emptyset$
    end if
    $len_{next} \leftarrow \text{len}(grid[altitude_i][x_i + 1]) - 1$
    $i_{min} \leftarrow \min(\max(y_i - MAX\_LAT\_VAR, 0), len_{next})$
    $i_{max} \leftarrow \min(y_i + MAX\_LAT\_VAR, len_{next})$
    for $i \leftarrow i_{min}$ **to** $i_{max} + 1$ **do**
      $points$.append($(x_i + 1, i, altitude_i)$)
    end for
    $altitude_i \leftarrow altitude_i + STEP$
  end while
  return points
end procedure

4.1.2 Weather Data Interpolation

Weather data is not obtained in a continuous format, but rather on a discrete grid with resolution $0.25° \times 0.25°$. To then obtain weather information at a points in-between grid steps, and also since it is rare for a point to lie exactly on a grid point, it is necessary to interpolate the weather data. As seen in Figure 4.4, this is a three step process. First, the temporal data is interpolated between the nearest two timesteps, provided in 1 hour intervals. Next, the data is interpolated vertically, between different pressure levels (hPa). For this project, three pressure levels are considered at 200hPa, 250hPa, and 300hPa. Finally, the data is then horizontally interpolated between the latitude and longitude where the final data can then be used for flight path calculations.

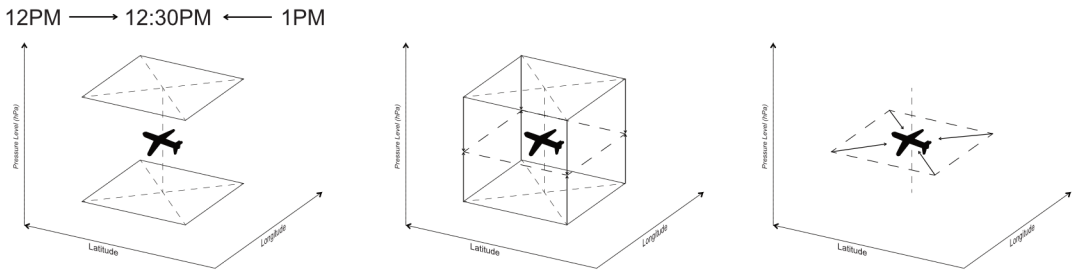


Figure 4.4: Weather interpolation based off the work by Mendoza et al. [26]

4.2 Flight Path Construction

Potential flight paths are constructed by starting at the departure point and selecting a point in the next step along the altitude grid that validates a set of constraints. Those constraints are:

- Within ± 15 deg points laterally
- Within $+4000 ft$ vertically

Both constraints are arbitrary, but are important in reducing the search space, as large changes in altitude and lateral position are unlikely between waypoints, due to being very fuel inefficient. Therefore, by placing a constraint on both, unrealistic solutions can be immediately discarded, as they would never be considered viable flight paths by airlines anyway.

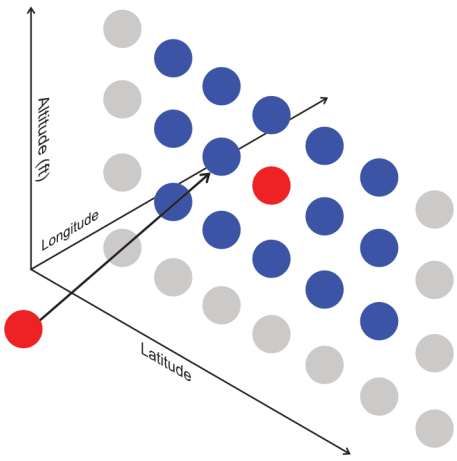


Figure 4.5: Flight path selection, with points on the geodesic path highlighted in red, and valid points to choose from highlighted in blue. The chosen path is shown with an arrow.

As can be seen in Figure 4.5, a point is chosen from points in the next step along the grid. This is repeated from the next node, until the destination node is reached, meaning the flight path has concluded.

Once the destination is reached, the aircraft's flight characteristics are calculated for each point along the flight path. These are calculated using the set of equations described in Section 2.3.

However, the CoCiP model requires small segment lengths between waypoints, yet candidate solutions produced through traversal of the routing graph have a significant distance between waypoints, due to the spacing between layers of the altitude grid. Therefore, a subset of flight

characteristics need to be calculated first, which then allows the flight to be resampled to 1 minute intervals. The resampled flight can then be ran against the full aircraft performance model, where emissions data is also included and calculated for the flight using *OpenAP* [38].

The full aircraft performance model structure is shown in Figure 4.6. It is important to note that the flight equations step will need to be ran twice if the flight is resampled. The first flight equations step is due to the time of arrival needing to be known at each waypoint for resampling to be effective. The second flight equations step is to recalculate speed, segment length and heading information for the newly interpolated data. Whilst this adds significant computational overhead it is necessary to enforce a small segment length constraint, otherwise the CoCiP model is almost completely ineffective.

Moreover, a nominal rate of climb/descent needs to be chosen for the resampling process. A value of 4.45 was obtained arbitrarily through experimentation, as this value produced results which most closely matched the altitude values of the comparison flight. This is an area that could be improved upon with better aircraft performance data, yet this is largely unavailable from aircraft manufacturers, and best estimates are frequently used within the larger flight optimisation community.

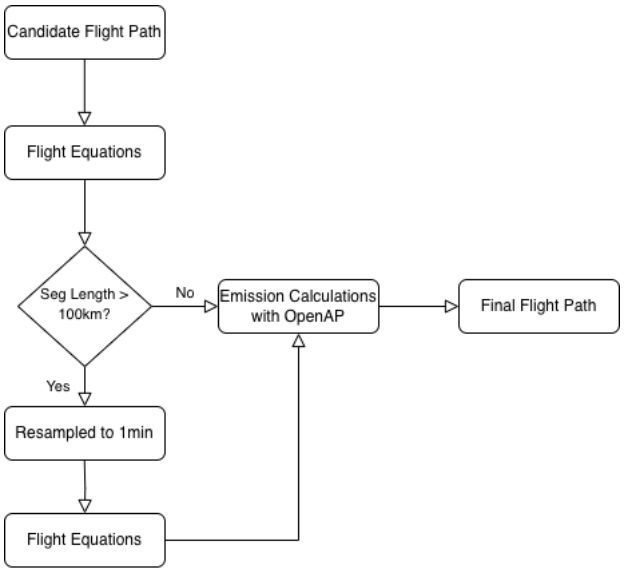


Figure 4.6: The order of operations in the aircraft performance model to calculate the equations of motion, and emission data.

This aircraft performance model architecture allows for a variety of flight paths to be modelled, even those not generated by an optimisation algorithm. Therefore, real flight data from *FlightAware* [13] can be passed through the model to obtain consistent performance data for comparison with flight paths computed through the optimisation algorithm.

4.3 Objective Function Definitions

Due to a multi-objective ACO algorithm being used, there are three cost functions that need to be defined. The first two are EF functions which considers both CO_2 emissions and the impact of contrail cirrus formation. The final one is the time function, which represents the total journey duration from the start of the cruise phase, to the end of the cruise phase.

Using this approach, it is also possible to incorporate other types of aircraft emissions with their own objective functions (such as NO_x [41]) using predictive models like LMDZ-INCA. However, that is outside the scope of this project.

4.3.1 Calculating Emissions

One approach to measuring the emissions from CO_2, is to use the amount of EF, as suggested by Teoh et al. [40]. The EF from CO_2 emissions can be estimated using the following equation.

$$CO_2EF[J] = \int_0^{TH} RF_{CO_2}\, dt \cdot S_{Earth} = [AGWP_{TH} \cdot (365 \cdot 24 \cdot 60^2)] \cdot TFC \cdot EI_{CO_2} \cdot S_{Earth}$$

where TH is the time-horizon (100 years, in line with the Kyoto protocols [1]), EI_{CO_2} is the emission index for CO_2 ($3.16 kg kg^{-1}$), S_{Earth} is the surface area of the Earth ($5.101 \times 10^{14} m^2$), $AGWP$ is an integration of CO_2 RF over a determined time horizon, and $AGWP_{100} = 92.5 \times 10^{15} yr\, W\, m^{-2}$ per $kg - CO_2$.

Using this equation, the CO_2 EF per kg of fuel burned is calculated to be $4.70 \times 10^9 J kg^{-1}$, meaning the total CO_2EF of a flight path (f_0) can be calculated as:

$$f_0(S) = (Fuel(S) \cdot CO_2EF)^{\omega_{CO_2}}$$

where ω_{CO_2} is the weighting of the CO_2 factor.

Although, alternative approaches are available, such as radiative forcing or the quantity of CO_2 in kg. For the purposes of this program, the kg of CO_2 released was chosen, as this reflects the immediate impact of a plane's flight path, rather than over a given time-horizon. Moreover, the European Environment Agency CORINAIR Manual [12] provides clear estimates for the total emissions produced per kg of fuel, allowing for simple estimations. Below is the calculation for a Boeing 737-400 for example.

$$f_0(S) = \left(\sum_{i=1}^{n} Fuel(i) \cdot 3.15 \right)^{\omega_{CO_2}}$$

This approach can be extended to incorporate emissions for several other greenhouse gases, such as NO_X and H_2O, whose EF impact is yet to be researched.

Heuristic Definition

For the heuristic function, a time estimation at the point is first calculated using an arbitrary speed value, and the distance from the departure point. A fuel flow estimate is then retrieved from a performance grid, pre-calculated for different aircraft types across several altitudes by the *pycontrails* [34] team. The fuel flow estimate is then used to calculate potential emissions by the *OpenAP* [38] emissions library.

4.3.2 Calculating Contrail EF

High quality contrail EF estimates can be produced using the CoCiP model [32], which simulates the size, lifetime and EF impact of contrails formed from inputted flight paths and weather information. The value outputted from the CoCiP model can be used by itself without modification.

However, due to long processing times associated with CoCiP, a rough estimation can be used during the optimisation process, which can be later refined through the use of CoCiP.

The rough estimation uses a pre-calculated CoCiP grid from the *pycontrails* [34] API, which is downloaded and interpolated to the routing grid. During the optimisation process, the contrail EF at each point along a candidate solution in the interpolated grid is summed together, creating an estimation of the contrail EF along that path.

The final contrail EF objective function is shown below:

$$f_1(S) = \left(\sum_{i=1}^{n} \xi(i) \cdot d(i) \right)^{\omega_{contrail}}$$

where $\xi(i)$ is the contrail EF estimation at a point in the routing grid, $d(i)$ is the segment length of the waypoint, and $\omega_{contrail}$ is the weighting of the contrail factor.

Heuristic Definition

The heuristic value for this objective function is similarly defined, however segment length is not taken into account. Therefore, the function is a single point lookup in the 4D contrail grid.

The value at all times in the grid is summed together to create the final heuristic estimate.

4.3.3 Time Function

The temporal objective function is then the difference between the final point in the path, and the initial point - representing the total time duration of the cruise phase of the flight.

$$f_2(S) = (t_n - t_0)^{\omega_T}$$

where t is the timestamp at a segment, n is the length of the path, and ω_T is the weighting of the time factor.

Heuristic Definition

The heuristic function for the time function is simply the expected time it would take to fly directly from the departure point to a specified node, at an arbitrary speed value.

4.4 m-ACO

4.4.1 Solution Construction

Based off the approach from Alaya et al. [17], each objective function has a separate pheromone and heuristic value at each edge. During solution construction, the ant will randomly select an objective to optimise at each step, using the respective pheromone information from the edge and corresponding heuristic values from the nodes.

4.4.2 Pheromone Update

Once solutions are constructed, the resulting paths are passed through each objective function, and the best path for each objective is stored, both globally and for the current iteration. Then, pheromones for each objective are updated according to the below formula:

$$\Delta\tau^i(c) = \begin{cases} \frac{1}{1+f_i(S^i)-f_i(S^i_{\text{best}})} & \text{if } c \text{ is a component of } S^i \\ 0 & \text{otherwise} \end{cases}$$

where f_i is the objective function being considered, S^i is the constructed solution and S_{best} is the best solution that has been discovered so far.

The probability of an ant selecting a neighbouring vertex is then given by:

$$p_S(v_i) = \frac{[\tau_S^i(v_i)]^\alpha \cdot [\eta_S^i(v_i)]^\beta}{\sum_{v_j \in Cand}[\tau_S^i(v_j)]^\alpha \cdot [\eta_S^i(v_j)]^\beta}$$

where $\tau_S^i(v_i)$ is the pheromone factor for objective function f_i, $\eta_S^i(v_i)$ is the heuristic factor for the same objective function and α and β are the weightings of these two factors, and can be adjusted to favour one factor over the other.

4.4.3 Solution Extraction

Unlike traditional ACO, m-ACO does not produce a single solution, but rather a Pareto optimal set of non-dominated solutions [17]. From this, an optimal solution for each individual objective can be extracted, or for all functions combined. This is especially useful for the problem space, as avoiding contrail formation is a conflicting objective with minimising the flight duration, meaning several solutions may be optimal simultaneously. Therefore, a user of the program can select which path they want to choose from the Pareto set.

Chapter 5

Implementation

The primary objective for the implementation was to encapsulate each section of the problem - allowing each to be individually developed and combined. Therefore, an iterative approach was used to first create a routing graph to be searched through, followed by the aircraft performance model and finally the implementation of the evolutionary algorithm. The encapsulation of each section of the program allowed for quick testing and iteration on each individual component, and for different approaches, APIs and algorithms to be experimented with.

5.1 Problem Space

5.1.1 Altitude Grid Construction

The core of the project is the altitude grid, which was based primarily off the approach described by Mendoza et al. [8, 9, 26]. It was immediately clear an altitude grid approach would be the most appropriate for the problem space, as alternative approaches, such as Bee Colony Optimisation [25], required more iterations and were less consistent with their results. Moreover, as the project developed, the altitude grid allowed for several optimisations to simplify and improve weather and flight characteristic calculations.

The altitude grid is constructed in three stages:

1. Geodesic Path Calculation
2. Routing Grid Construction
3. Altitude Grid Construction

Initially, each stage mirrored the process described in Section 4, however as the project developed it was clear that the staggered nature of the routing grid was problematic for the construction of valid flight paths, specifically from the departure side of the routing grid. As seen in Figure 5.1, the position of the 0 index in each layer of the grid, would change in location with each step through the grid, making it exceptionally difficult to calculate valid consecutive points. An approach could be taken to introduce extra constraints, checks and conditions to determine valid consecutive points, so that the staggered grid could be preserved.

However, it was deemed that an alternative approach could be used to preserve functionality, whilst being significantly simpler. Instead of a staggered grid, the layer size now stays constant from the departure side, and only shrinks as the grid approaches the destination. The same points are accessible to the aircraft, and the consecutive points algorithm means that the newly added points are still inaccessible, as shown by the greyed out points in Figure 5.1.

Although, this approach significantly increases the number of points that need to be included in the routing graph calculations later - which subsequently increases processing time. This was mitigated through storing the routing graph locally, so that the calculation only needs to be performed once, as well as by reducing the number of layers in the routing grid. Mendoza et al. [26] used a much higher layer count, as well as extra constraints to ensure smooth flight paths. However, this meant that optimisation took much longer as there were many more nodes that needed to be included in the search. Instead, a lower layer number is used in conjunction with resampling to ensure smooth flight paths, without losing on data points. However, this comes with the drawback that generated paths are potentially worse, although this was not experienced in practice.

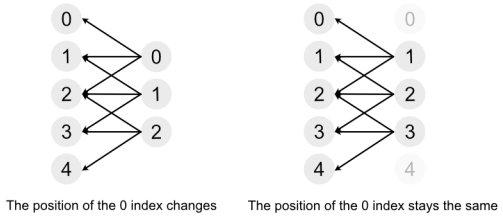


Figure 5.1: The routing grid implementation keeps the 0 index consistent from the departure side.

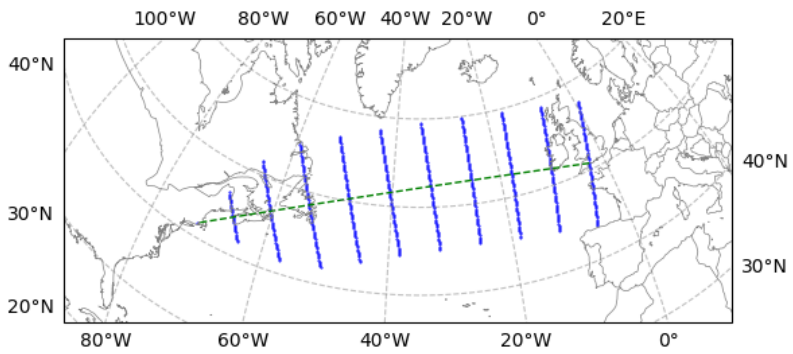


Figure 5.2: The final routing grid implementation, with each layer in blue and the geodesic path in green

To turn the routing grid into an altitude grid, points are added at $2000ft$ intervals from $31000ft$ to $39000ft$.

5.1.2 Routing Graph Construction

Initially, potential flight paths were constructed by checking whether index values were within $\pm\alpha$ of the current index, where α is an arbitrary lateral offset variable. However, this proved slow and made it difficult to store pheromone and heuristic information later.

Therefore, a unidirectional *NetworkX* [16] graph was constructed to store all possible edge connections from every single node in the altitude grid, as well as allowing for data to be stored at both the edges and nodes, which was eventually used to store pheromone and heuristic information respectively.

To construct the graph, a consecutive points algorithm, shown in Figure 5.3, was ran for each point in the altitude grid. An edge is then drawn between the current node, and all consecutive nodes.

After implementing the routing graph, neighbour look-ups during optimisation were much quicker, and secondary look-ups to a different data structure were no longer needed, as all relevant information on edge and node values were stored directly as part of the graph.

To prevent the need to re-calculate this graph for each execution of the program, the result

is stored as a GML (Graph Modeling Language) file, and retrieved at program initialisation.

```
def get_consecutive_points(
    xi,
    yi,
    altitude,
    grid,
    max_lateral_var=config.OFFSET_VAR,
    max_altitude_var=config.MAX_ALTITUDE_VAR,
    altitude_step=config.ALTITUDE_STEP,
):
    max_alt = altitude + max_altitude_var
    max_alt = max(config.STARTING_ALTITUDE, min(max_alt, config.MAX_ALTITUDE))

    points = []
    current_altitude = altitude
    while current_altitude <= max_alt:
        if xi + 1 == len(grid[current_altitude]):
            return None
        next_layer_length = len(grid[current_altitude][xi + 1]) - 1
        min_i = min(max(yi - max_lateral_var, 0), next_layer_length)
        max_i = min(yi + max_lateral_var, next_layer_length)
        for i in range(min_i, max_i + 1):
            points.append((xi + 1, i, current_altitude))
        current_altitude += altitude_step

    return points
```

Figure 5.3: The consecutive points algorithm, used to determine which points in the next layer are accessible from a given node

5.2 Aircraft Performance Model

Throughout the course of implementation, the flight path construction method went through several iterations. Since this was the primary bottleneck during each iteration, a lot of the optimisation focus was spent investigating marginal performance gains. However, this led to a difficult trade-off between performance and model accuracy. Therefore, the two approaches had to be carefully balanced and tweaked through frequent experimentation to achieve optimal results.

The key areas of the performance model that had room for optimisation are listed below:

- Contrail Formation Model
- Fuel Flow Model
- Weather Model

The initial approach taken had a primary focus on accurately modelling the flight space, as this was deemed necessary to provide useful outputs. However, a purely accuracy focused

approach led to unreasonably long iteration times which would go against the performance requirement of the program. Therefore, compromises had to be made for each model to balance both requirements.

5.2.1 Contrail Formation Model

A full CoCiP simulation provides an accurate model of how contrails develop from a flight path over time, as well as their potential EF impact, making it a useful tool to determine which flights are causing highly warming contrails. As such, the initial version of the performance model used a full CoCiP simulation for every trajectory generated, providing better contrail estimations than using the Schmidt-Appleman/Schumann criterion (SAC) [3, 33] to detect and avoid ISSRs.

Although, as the project progressed it was clear that the CoCiP prediction model [32] needed to be replaced with a 4D contrail EF grid, since the CoCiP simulation took several seconds per simulation, and hundreds of simulations needed to be ran during the optimisation process. The *pycontrails* API [34] provides a pre-calculated grid that is averaged across a range of azimuth values. The impact of the azimuth value on contrail warming is an active area of research, meaning that the impact of an averaged value on the model accuracy is unknown. Regardless, the *pycontrails* team found a gridd approach to return EF values within 20% of those given by the CoCiP simulation. Whilst this is a significant difference, the grid still provides a useful heuristic for detecting highly warming contrails, and is at the very least still a better approach than SAC.

The pre-calculated grid is downloaded locally for the routing-grid region, and for 12 hours after the flight departure time. The grid is then interpolated to the routing grid points, and is used during the optimisation process to determine a Pareto set of optimised flight paths. A full CoCiP simulation is ran on the Pareto set, as well as the original flight path for an accurate model of the contrail EF impact over the contrail's lifetime.

5.2.2 Fuel Flow Model

The fuel flow model similarly went through several iterations throughout the course of the project. Initially, the *OpenAP* model was used, but a simpler, more-performant approach was trialled. A nominal fuel flow value was used, acting as an average of the fuel flow over the whole flight path. However, this approach had significant disadvantages, such as proportionally linking the CO_2 and time objective functions. Effectively, this reduced the number of objective

functions from 3 to 2. In reality, these are closely-linked, but separate objectives and this should be reflected in any performance model.

Therefore, the decision was made to revert back to using the *OpenAP* flight model to provide accurate fuel flow estimations. The computational costs associated were able to be counter-balanced through savings from multi-processing later described in Section 5.5. This approach allows for extra emission calculations, based off data from the European Environment Agency's CORINAIR manual [12], to be added to the objective function (such as H_2O, NO_X, CO, SO_2 and HC) which will be discussed later in this report.

5.2.3 Weather Model

The weather model was at first stored in an interpolated grid using locally cached data from the ECMWF ERA5 Dataset [7]. During the optimisation process, the relevant data point in the dataset was then re-interpolated to the time at the current waypoint. This meant that timestamps were available for when the aircraft would be at that point, providing an accurate estimation of the weather.

However, continuously accessing and re-interpolating the dataset was slow, especially over the large number of iterations that ACO requires. Therefore, an approach was taken to pre-calculate temperature and wind data at each node in the routing graph, allowing it to be quickly accessed during optimisation. To perform this, time of arrival estimates at each node were calculated based on their distance from the departure point, using the nominal thrust value. Then, the weather grid would be interpolated to these points and stored as part of the routing graph, which can be quickly accessed by the performance model.

In practice, this approach proved to be only marginally faster than directly accessing the dataset from the performance model and, in cases, the resulting paths were worse, prompting the decision to revert back to directly accessing the ERA5 Dataset.

5.3 ACO Algorithm

Several different ACO algorithms were experimented with, primarily: traditional ACO [36], MAX-MIN Ant System [37], and m-ACO [17] which is an extension of MAX-MIN for multi-objective problems.

Both ACO and MAX-MIN use a single objective function and pheromone structure to guide the optimisation process (although MAX-MIN clamps pheromone values to encourage more

exploration). Therefore, to optimise for multiple objectives, a function needs to be created that equally weighs every objective. There were several iterations of an overall objective function that attempted to normalize the EF_{CO_2}, $EF_{contrail}$ and the flight duration τ under a single function. One iteration of a unified objective function is shown below.

$$f(x)_{\text{total}} = \left(\frac{EF_{CO_2}}{1 \times 10^6}\right)^{\omega_{CO_2}} + \left(\frac{EF_{contrail}}{1 \times 10^{18}}\right)^{\omega_{\text{con}}} + \left(\frac{\tau}{10}\right)^{\omega_\tau}$$

However, the denominator values were largely chosen arbitrarily resulting in sub-optimal, and difficult to tweak, results. Moreover, optimising for several objectives at once meant that the overall solution was poor, which is confirmed by Alaya et al. [17]. As such, a multi-objective approach was necessary to generate a Pareto set of optimal flights, where specific objectives could be prioritised over others.

As described in Section 4.4, the m-ACO algorithm optimises several objectives at once through the use of several ant colonies and/or pheromone structures. The chosen implementation uses a single colony, and several pheromone and heuristic structures, corresponding to each objective function.

For each step of solution construction, the ant will select an objective to optimise, using the relevant pheromone and heuristic values for that objective in the probability calculations. After solutions are generated, an elitist approach inspired by MAX-MIN [37] is used to update pheromone structure for each objective function; where only the edges from the best solution, from that iteration are deposited with more pheromones.

During the optimisation process, a Pareto set of non-dominated solutions is stored, and a solution along the Pareto front can be selected by the user. Solutions from a Pareto set proved to be always better in one objective than solutions produced from a combined objective function, and often better over all objectives, meaning the algorithm ultimately proved to be the appropriate choice for the project domain.

5.4 Thrust Variation

There were several attempts to include thrust variation as part of the optimisation process, meaning that the aircraft could either pick a different thrust value between waypoints. Another approach was also investigated, where the aircraft would pick an average thrust value for the entirety of the cruise flight segment.

However, both approaches were ultimately discarded. The first approach required a second

ACO optimisation process, using the method described by Mendoza et al. [26], to search a graph of varying MACH values at each waypoint along the initial flight path. This effectively doubled the program execution time, making it exceedingly difficult to try and meet the total runtime requirement.

The second approach proved to be almost non-existent, as the algorithm would always pick the highest available average MACH value for the flight path, as this would almost always provide the best results for all three objectives.

Therefore, generated flight paths were instead assigned a nominal thrust value of MACH 0.84. This value was chosen, as when applied to the chosen real flight path, it provided the closest estimate of the actual time at the end of the cruise phase.

5.5 Multiprocessing

One of the key advantages of ACO, is that it is exceptionally simple to implement multi-processing. This was an important area for improving the performance of the program, to meet the computation time requirement.

Each ant runs on an independent process in parallel. Once all ants have completed solution construction and objective evaluation, they are compared against the Pareto set of solutions, which is accordingly updated.

Out of the optimisations described in this section, this change had the most substantial impact on performance, at relatively little technical complexity, allowed by the nature of evolutionary algorithms.

5.6 Modularisation

To further separate individual components and encapsulate functionality, several independent modules were created within the project, which each were able to be accessed and applied through a single entry point. Each module is listed below, alongside its functionality within the program:

Module	Functionality
`aco`	Ant Colony Optimization algorithm implementation
`display`	Functionality related to displaying results or visualizations
`objectives`	Implementation of objectives used in optimization
`performance_model`	Functionality related to modelling the problem space
`routing_graph`	Implementation of the routing graph data structure
`utils`	Utility functions or helper methods for file and unit conversion

Table 5.1: Functionality of Each Module

The relationship between components, and their containing modules are showcased in Figure 5.4.

Each module is a self-contained unit of functionality, that can be individually maintained. By structuring the program like so, it was simple to develop and integrate new systems. For example, adding new objective functions would be simple, as a new objective component could be added to the objectives module, and the program would now consider an extra objective during optimisation.

5.7 Iteration Limitations

During development, it became clear that there was a soft limit on the number of iterations that could be performed by the optimisation algorithm. This number ranged from 300-500, and upon further investigation it was clear there was an issue with large amounts of memory being allocated to the program (∼250GB). Eventually, the issue was traced to aircraft data being opened continuously, but never being closed by the *OpenAP* library [38]. Therefore, I submitted a GitHub pull request [1] to attempt to address this issue, however at the time of writing the pull request is still open and unaddressed. Unfortunately, this meant a maximum of 200 iterations could realistically be performed before the program would grind to a halt.

5.8 Testing

To ensure program functionality throughout development, extensive unit tests were implemented for each module. This ensured each individual piece of the program retained intended functionality as new features were built out during development. The Python `unittest` library is used to create the unit tests. Alongside automated unit tests, extensive manual testing was

[1] The open PR addressing the memory issue: `https://github.com/junzis/openap/pull/46`.

performed to catch cases where behaviour was unexpected. For example, a valid flight path may have been generated, but it may be much slower than expected. Cases like these are hard to catch with unit tests, especially due to the stochastic nature of ACO.

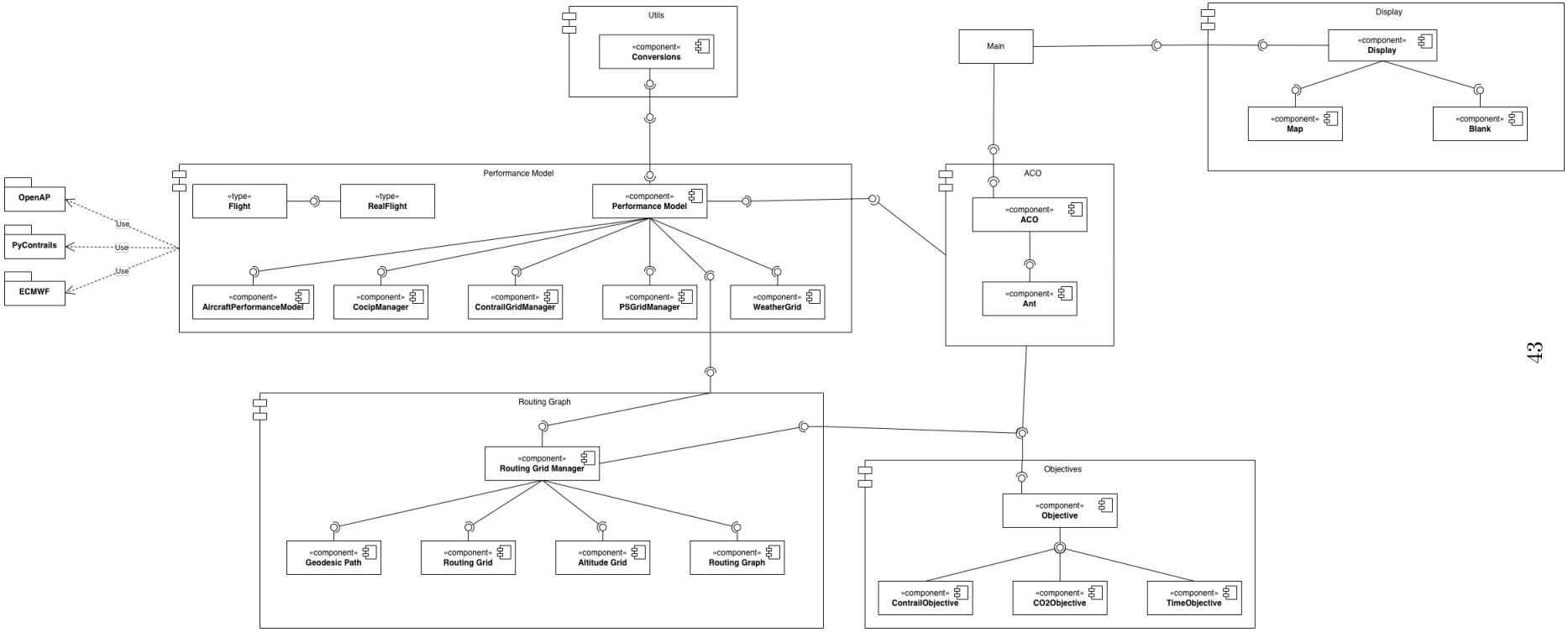


Figure 5.4: The full component diagram showing how components are split between modules.

Chapter 6

Legal, Social, Ethical and Professional Issues

6.1 Ethical Concerns

Whilst, the primary intention of this project is optimising flights to avoid contrail cirrus formation, there are a number of ethical and professional concerns as part of the project domain, and possible extensions to other domains. For example, the motivation of the project is to help mitigate some of the impact of climate change, however this problem space requires extra care to be taken to ensure that any attempts to reduce global emissions are successful. Otherwise, there is the concern that an unsuccessful attempt may instead increase global emissions. For example, it may be the case that the optimisation algorithm successfully reduces contrail cirrus, but it may also inadvertently increase CO_2 emissions, negating or possibly outweighing any benefits, ultimately causing more harm than good.

Moreover, there is the professional concern of adhering to global aviation standards and practices, in order to avoid extra strain on flight scheduling and global air traffic. One of these concerns are conventions on whether planes fly at odd or even altitudes depending on their cardinal direction, which need to be followed by any program creating flight paths. Otherwise, any potential solutions will be essentially useless in a real-world setting.

There is also the ethical concern of the potential of this projected to be extended to non-commercial, military domains. Some nearby research uses similar methods to pilot autonomous drones to avoid hostile ordinance [30, 46], which has historically been used to target civilian populations in several global conflicts. The intention of this project is not to create a model

that reflects this, however it can easily be conceived of by following a similar method described.

6.2 British Computing Society Code of Conduct & Code of Good Practice

In compliance with the British Computing Society (BCS) Code of Conduct and Code of Good Practice, this project respects the authority and guidance of global aviation bodies, as well as adhering to professional standards for the development of software.

Chapter 7

Experimentation

7.1 Parameter Selection

Whilst different parameters were experimented with during testing, a set of base parameters were used as a control, and are to be assumed as the parameters used during optimisation unless stated otherwise. They are listed in Table 7.1.

Constant	Value
# of iterations	100
Evaporation Rate	0.5
# of layers	10
# of ants	8
MACH	0.84
Starting mass	240,000kg
Aircraft Type	B77W
Starting Altitude	31,000ft
Maximum Altitude	39,000ft
Contrail Weight	1
CO_2 Weight	1
Time Weight	1
τ_{min}	0
τ_{max}	1
α	2
β	1

Table 7.1: Constants

7.2 Comparison to a Real Flight and Random Flights

Results generated by the optimisation algorithm were compared against two flights. Firstly, a random flight was generated along the same routing graph, where a valid consecutive point was

selected at random until the destination node was reached. The aircraft performance model was then ran against this flight to generate flight characteristics, emissions data, and contrail lifetime information. If the Pareto set has multiple flights, the average value from all flights in the Pareto set is used as the objective value. The raw data for each Pareto set is found in the Appendix.

It is also important to note that for the purposes of comparison, a single random flight is used for reference per investigation, rather than per optimisation. This path was chosen using a random number generator to select a flight from the set of all random flights, to provide a fair basis for comparison.

Likewise, the same performance model was ran for a real flight path, obtained from *FlightAware* [13], from London Heathrow to John F. Kennedy airport, on January 31st 2024. The flight was trimmed to only the cruise phase of the flight (from the top of climb, to the top of descent), and resampled to 1 minute intervals. The flight had missing data from when the plane was over remote sections over the Atlantic ocean, so the path was interpolated to fill the missing data points. The flight is shown in Figure 7.1.

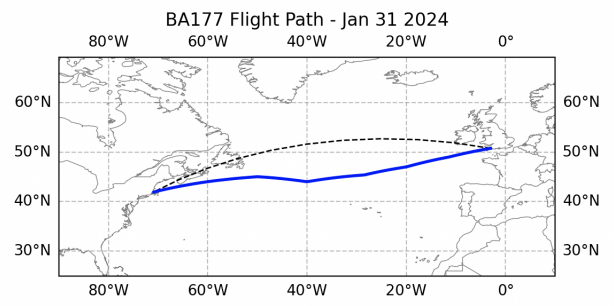


Figure 7.1: The flight path taken by BA177 on January 31st 2023 in blue. The geodesic path is shown in black.

7.3 Preface on Result Presentation

7.3.1 CoCiP Objective

In the following results, the objectives are defined using the same objective functions specified in Section 4.3, with the exception of the "CoCiP" objective, which is simply the sum of the EF estimation generated by the CoCiP model for the specified flight path. Generally, this value will be a better EF estimate than the one provided by the "Contrail" objective, but it is not optimised for by the algorithm, due to the reasons and requirements discussed in Section 5.2.1.

Both values are included in the results tables though, to show the limitations of the 4D contrail grid in estimating the formation of contrail cirrus.

7.3.2 Hypervolumes

To compare the quality of a Pareto front, the hypervolume of dominated solutions will be compared between runs, using the approach described by Zitzler et al.[50]. A Pareto front with a larger hypervolume is considered better, due to it dominating a larger set of solutions. Due to some objectives having scales orders of magnitude larger than others, it is necessary to normalise the values of the Pareto set. This avoids the hypervolume being biased towards certain objectives in the raw value space. The values are normalized to within the ideal and nadir points, using the *pymoo* [6] library.

As for the arbitrary reference point, this is defined as slightly beyond the nadir point, so that it is dominated by the Pareto set. Since the data has been normalised to within 0 and 1, a value of $(1.1, 1.1, 1.1)$ is used, as suggested by Wang et al. [45] and Zapotecas-Martínez et al. [49]. There has been further research into the specification of the reference point and its impact on the hypervolume, however that is outside the scope of this project.

7.4 Single Path Comparison

Before experimenting with different parameters or objectives, it is first important to establish the ability of the optimisation algorithm, using experimentally chosen hyper-parameter values. These values were determined largely through trial-and-error, although a grid search implementation should be considered for future extensions.

A flight was randomly selected from the Pareto set to be compared against both reference flights as the optimised flight path. In practice, any flight along the Pareto front can be selected.

Figure 7.2 and Table 7.2 show the flight path with contrails overlayed, and the raw objective values respectively. It is clear that the optimised flight path is a significant improvement over both reference flights. The optimised flight path is more fuel efficient, quicker overall, and produces far less contrail cirrus. This is a major improvement over both reference flights, showcasing that the algorithm works well in optimising all three objectives.

Solution	Contrail	CO2	Time	CoCiP
Real Flight	1.56E+15	6.11E+06	6.35	1.20E+16
Random Flight	8.36E+14	6.80E+06	6.68	7.15E+15
Optimised Flight	4.75E+13	5.75E+06	6.13	1.31E+14

Table 7.2: A randomly selected path from the Pareto set compared against the reference flights.

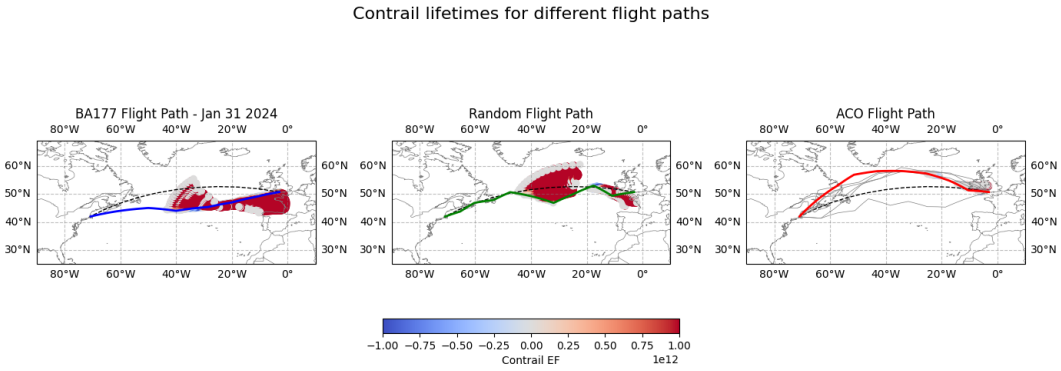


Figure 7.2: The flight path generated optimising for all objectives, compared to a random path and the flight from BA177.

7.5 Single objective vs Multiple objectives

As the algorithm is multi-objective, results from each individual objective are unlikely to be globally optimal, especially when compared against an algorithm that solely optimises one objective. Therefore, as an initial point of comparison the impact (and potential loss) from having to compromise was investigated for each objective.

	BA177 Diff				Random Path Diff			
Objective	Contrail	CO2	Time	CoCiP	Contrail	CO2	Time	CoCiP
Contrail	N/A	7.03%	4.94%	-2849.46%	N/A	-1.67%	0.45%	-57.58%
CO2	32.64%	-7.23%	-3.25%	-416.98%	81.29%	-17.26%	-8.13%	72.38%
Time	-364.36%	-5.44%	-2.70%	-529.59%	-28.96%	-15.30%	-7.55%	66.36%
All	-1323.45%	-0.80%	-0.16%	-2033.34%	-295.33%	-10.23%	-4.89%	-13.98%

Table 7.3: Optimising for single objectives vs multiple objectives [1]

Solution	Hypervolume
All objectives	1.24
Contrail	0.02
CO2	0.11
Time	0.93

Table 7.4: Hypervolumes for each Pareto front with different optimising objectives

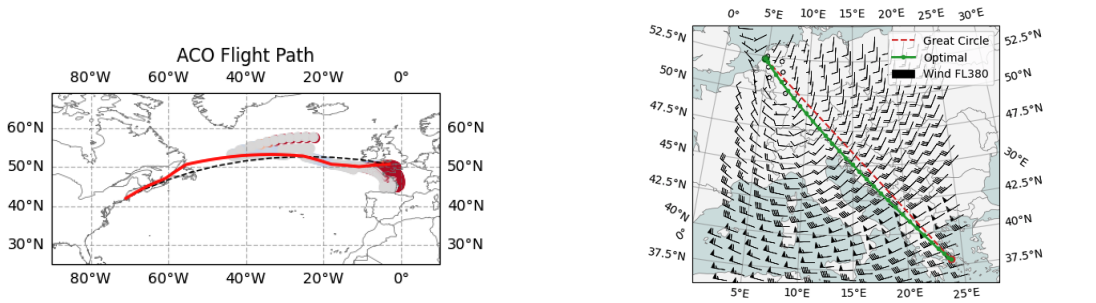
As can be seen in Table 7.3, the path that aims to optimise all objectives is able to improve on both reference flights for all objectives. Although, it does not provide the globally optimal

[1]Some values are N/A, due to either value being 0 when finding the percentage difference between them.

path for any single objective, which was expected before the experiment. However, it is clear in Table 7.4 that by optimising for all objectives, the hypervolume is much larger than for single objectives, meaning that the algorithm is successfully balancing all three objectives to create dominant solutions.

It is also clear that solely optimising for a single objective creates flight paths that perform worse on other objectives compared to the reference flight, but are significantly better on the single objective they aim to optimise. This matches the intuition, as the ants are only looking to improve on that objective, which is especially important when objectives are potentially conflicting.

For example, the colony only seeking to reduce CO_2 emissions, aims to take the shortest path along the geodesic line between the two points, as can be seen in Figure 7.3 . This produces a similar result as one that would be output from a traditional optimiser like *OpenAP* [38] that simply takes into account the wind vectors and the shortest path between the two points.



(a) The flight path generated whilst trying to reduce CO_2 emissions.

(b) An example flight path using the *OpenAP* trajectory optimiser, from the *OpenAP* website.

Figure 7.3: Comparison of a path only optimising to reduce CO_2 vs. a traditional optimisation process.

However, the colony only optimising to avoid contrail formation zones will carefully route to avoid them, even if this means an inefficient flight path at a low altitude. Figure 7.4 shows the generated path that almost completely avoids the contrail formation zones from the 4D contrail grid.

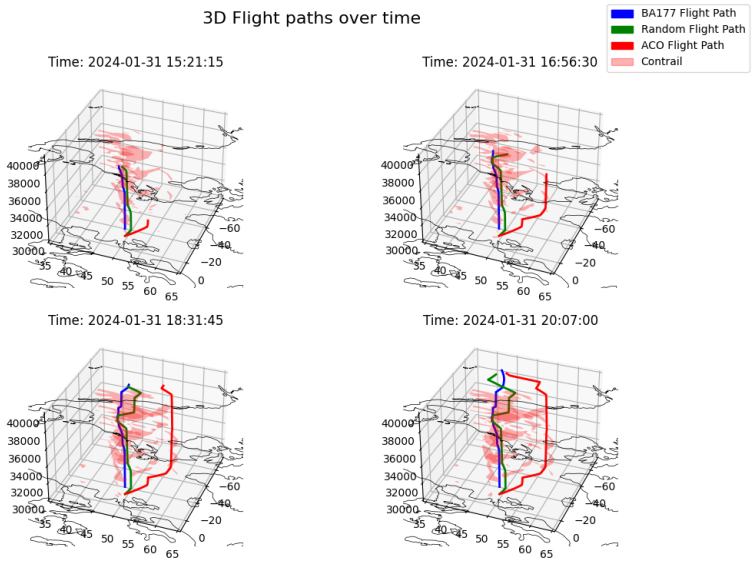


Figure 7.4: The path generated to avoid contrail formation zones, at 4 different timestamps.

7.6 Hyper-Parameter Investigations

Several different hyper-parameter values were investigated for the optimisation algorithm, in order to find out how their value could impact the optimisation process. These include the:

- Iteration Count
- Evaporation Rate
- Number of Ants

These parameters were largely chosen arbitrarily, as they were believed to have the biggest impact on the optimisation process. Further research however could investigate other parameter's impact however, such as objective weighting and the heuristic vs pheromone tradeoff.

7.6.1 Iteration Count

The first hyper parameter that was experimented with was the number of iterations ran by the optimisation algorithm. A maximum of 200 iterations was determined for the reasons outlined in Section 5.7. Each run was separate, meaning all graphs and values were re-initialised between runs. This was deemed most appropriate to attempt to isolate only the impact of the iteration count, compared to collecting data of the current Pareto set at each iteration count during a single 200 iteration run.

Table 7.5 and 7.6 showcases the results from runs with different numbers of iterations. As expected, a lower number of iterations will produce worse results overall, although it may be improve on certain objectives if the a single ant is able to get "lucky" when constructing a path. However, there appears to be a limit in the quality of solution, which is generally found around 50 iterations, with the size of the hypervolume diminishing after this point. This may be due to a variety of reasons, such as stagnation, where the algorithm has exhausted the search space and it begins to exploit sub-optimal solutions more.

	BA177 % Difference				Random Path % Difference			
Iterations	Contrail	CO2	Time	CoCiP	Contrail	CO2	Time	CoCiP
1	-191.22%	4.31%	2.43%	-740.38%	32.87%	3.42%	1.41%	61.79%
10	-1097.56%	2.95%	1.72%	-1858.05%	-176.06%	2.04%	0.69%	10.98%
50	-350.31%	-2.21%	-0.79%	-5416.97%	-3.80%	-3.16%	-1.85%	-150.82%
100	-11658.31%	1.83%	1.64%	-38224.53%	-2610.47%	0.91%	0.60%	-1642.35%
200	-2108.45%	-2.18%	-0.34%	-2318.15%	-409.08%	-3.13%	-1.39%	-9.94%

Table 7.5: Impact of iteration count on values

Solution	Hypervolume
1	0.38
10	0.73
50	1.46
100	1.41
200	1.35

Table 7.6: Hypervolumes for each Pareto front with different numbers of iterations

Shown Path	Contrail	CO2	Time	CoCiP
1 iteration	7.05E+14	6.14E+06	6.37	2.09E+15
200 iterations	8.94E+12	5.89E+06	6.25	1.98E+12

Table 7.7: Objective values for the shown flight paths in Figure 7.5

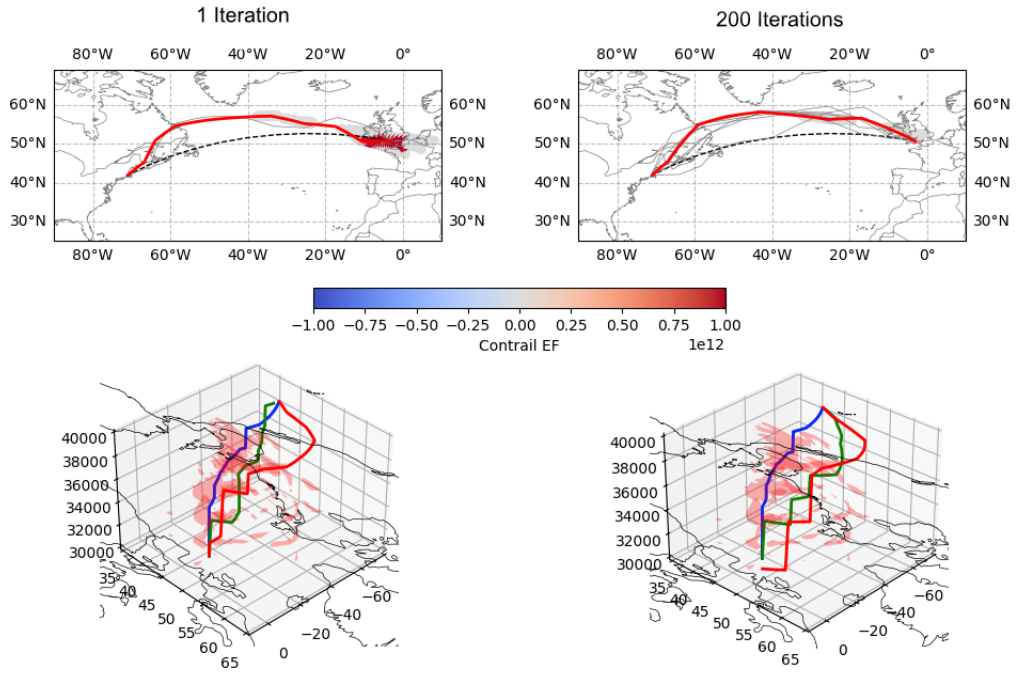


Figure 7.5: Flight paths after 1 vs 200 iterations, with red areas indicating contrail formation zones in the 4D contrail grid

Figure 7.5 shows two flight paths, the first generated after only a single iteration, and the second after 200 iterations. Interestingly, both paths largely take a similar route, suggesting a strong heuristic guiding initial solution construction and exploration. However, by looking at Table 7.7, it is clear the second path is able to perform significantly better. This demonstrates that the path is refined through more iterations, and that the optimisation algorithm is working.

7.6.2 Evaporation Rate

The next hyper-parameter that was investigated was the evaporation rate, which controls how much pheromone is maintained between iterations. For example, a value of 0.2 means 20% of the pheromone evaporates away between iterations.

Table 7.8 and 7.9 shows the results from runs with evaporation rates of 0.2, 0.5 and 0.8. From this it is clear that a careful balance is required between exploration and exploitation in order to achieve the best results, with a value of 0.5 providing the largest hypervolume. Firstly, in the case of evaporation being too high, ants could be initially exploring "bad" paths that it continues to exploit, due to low pheromone values on alternative paths. Similarly, an evaporation value that is too low may provide the ant with too many choices, leading it to explore too much, rather than exploiting the knowledge it currently has of the search space.

Figure 7.6 nicely demonstrates this, where the Pareto front for a colony with a balanced evaporation rate is spread out, covering a much larger volume of dominated solutions.

	BA177 Diff				Random Path Diff			
Evap. Rate	Contrail	CO2	Time	CoCiP	Contrail	CO2	Time	CoCiP
0.2	-249.44%	-0.30%	0.17%	-1926.99%	-752.30%	-8.90%	-4.01%	-678.55%
0.5	-183.01%	-1.71%	-0.76%	-1148.70%	-8.41%	-9.33%	-4.46%	-5.75%
0.8	-375.77%	-2.51%	-1.24%	-628.30%	-82.25%	-10.19%	-4.96%	38.32%

Table 7.8: Results from changing the pheromone evaporation rate

Solution	Hypervolume
0.2	1.31
0.5	1.40
0.8	1.19

Table 7.9: Hypervolumes for each Pareto front with different evaporation rates

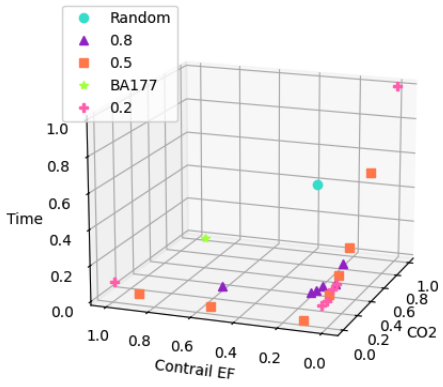
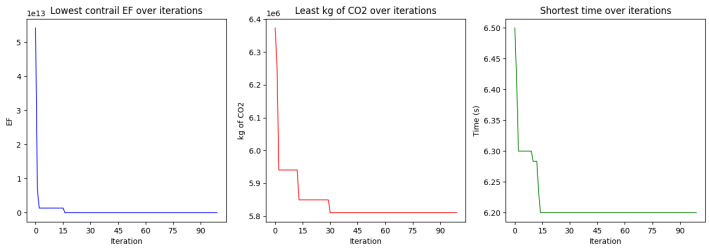
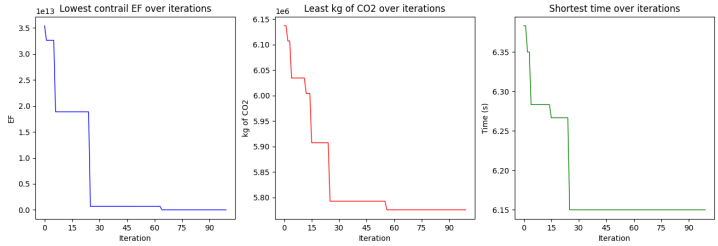


Figure 7.6: The Pareto front for each run with different evaporation rates.

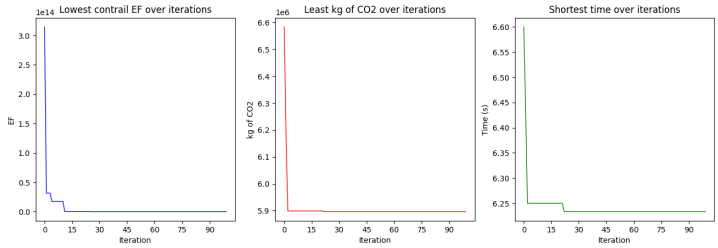
The trade-off between evaporation rates becomes clearer when looking at how much of the search space has been explored. Figure 7.7 provides a proxy for this, by plotting the best objective value found for each objective, over time. This roughly correlates to how much of the graph the ants have explored, although this is an area for further exploration in the future. From the figure, it is clear that an extremely high or low evaporation rate results in the algorithm reaching a plateau very quickly, which it is unable to leave throughout the rest of the run. Despite this, a lower evaporation rate is still able to find better values due to the extra exploration freedom. An evaporation rate of 0.5 though is clearly the best compromise between exploration and exploitation, with the colony able to leave the plateau and find the lowest values for each objective out of the three runs.



(a) Evaporation rate of 0.2



(b) Evaporation rate of 0.5



(c) Evaporation rate of 0.8

Figure 7.7: Global best objective values with evaporation rates of (a) 0.2 (b) 0.5 and (c) 0.8

7.6.3 Number of Ants

The final hyper-parameter that was investigated was the number of ants in the colony. Tables 7.10 and 7.11 show significant improvement between 1 and 8 ants, with a smaller but noticeable improvement from 8 to 16 ants.

	BA177 Diff				Random Path Diff			
No. of Ants	Contrail	CO2	Time	CoCiP	Contrail	CO2	Time	CoCiP
1	-887.69%	-0.26%	-0.08%	-1621.39%	-106.76%	-8.85%	-4.80%	100.19%
8	-1985.32%	1.27%	1.11%	-4291.19%	-336.54%	-7.19%	-3.56%	100.47%
16	-13520.91%	-4.26%	-1.60%	-36284.70%	-2751.38%	-13.19%	-6.40%	103.93%

Table 7.10: Results from changing the number of ants in the colony

Solution	Hypervolume
1	1.00
8	1.30
16	1.44

Table 7.11: Hypervolumes for each Pareto front with different numbers of ants

Naturally, it is expected that more ants exploring the search space should result in better solution. It is therefore a good indicator of the algorithm's success that the intuition is reflected in the collected results.

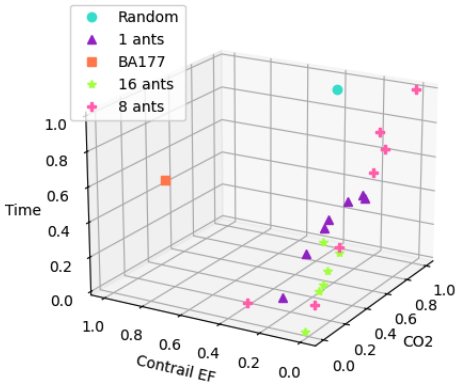


Figure 7.8: The Pareto front for each run with different numbers of ants.

It is also interesting to note that the variation in the Pareto front is much smaller for the run with 16 ants, as shown by the close grouping of the green stars near the $(0, 0, 0)$ point in Figure 7.8. This implies that the algorithm was approaching a convergence point. However, this may also attributed to random variation between result runs, as the run with 8 ants has far more variation in the Pareto front than the run with a single ant.

Chapter 8

Evaluation

It is clear from experimentation that the program is successful in the primary aim of optimising a flight path in order to avoid contrail formation. Moreover, the project has exceeded initial expectations, by also improving upon reference flight paths on all objectives. When optimising for single objectives, reductions of 2.70% and 7.23% were observed for the flight duration and total CO_2 emissions, with total contrail EF being reduced by 2849.46%. For multiple objectives, reductions of 0.79%, 2.21%, and 5416.97% were observed for the same objectives respectively.

8.1 Comparison to Other Works

However, it is hard to compare this reduction to other optimisation approaches, where different objective measures were used. For example, Mendoza et al. [26] used a flight cost to approximate the fuel burn and flight duration objectives. On a 3D grid, comparable to the one used in this project, they observed a reduction in flight cost of 6.82%. An alternative A* approach by Vergnes et al. [44] saw the majority of their performance gains from the path increasing in altitude above the reference flights. These optimisation gains were significantly reduced when the altitude was adjusted to match the reference trajectories. Also, neither paper considers the implications of contrail cirrus formation, although both do reference it as potential future extensions.

Lim et al. [23] uses a similar multi-objective approach to optimise for contrail reduction, however their approach combines several objectives into a single objective function, with weightings for each individual component. As found by Alaya et al. [17], this generally provides worse results than by optimising for separate objectives. There is no comparison to a real flight tra-

jectory too, making it hard to make comparisons between either approaches efficacy. Teoh et al. [40] instead employed a simpler contrail avoidance technique with minimal altitude tweaks to a selection of flight paths, proving effective across a fleet of aircraft with a 59.3% reduction in contrails with a 0.014% increase in fuel consumption. However, like many similar approaches, this compromises on other objectives to reduce contrail formation.

8.2 Project Limitations

It needs to be said however, that the results observed from this project were against a single flight that was selected due to it being particularly warming and inefficient, making it an ideal candidate for optimisation. Due to the complexity of the problem, further testing is required to confirm these results against several flights, or potentially fleets of aircraft, that was not afforded within the scope of this project.

Moreover, it proved challenging to accurately model the flight characteristics of the aircraft, with many considerations that need to be made around units and weather factors. Therefore, as much of this responsibility was outsourced to libraries such as *OpenAP* [38] and *pycontrails* [34], to try and ensure accuracy as much as possible. There is still plenty of room for improvement in this area though to ensure complete accuracy.

There are also several concerns around the performance of the program, with 100 iterations taking 30 minutes to complete. Whilst this is within the initial requirements of the program, there is still room for improvement, especially in a real-world setting where this would need to be ran for many flights, with changing weather situations. This benchmark is from a run with a pre-computed routing graph too. For a run where a new routing graph was constructed, the benchmark was closer to 3 hours, although as mentioned this only needs to be calculated once.

During the course of this project, there has been further research surrounding the harmfulness of contrails by Lee et al. [21], with the suggestion being that contrails are not as harmful to the environment as previously predicted. This is still unknown however, and is an active area of research. Due to a definitive answer not being known, this project was undertaken with the assumption that contrails are harmful to the environment. However, the flexibility of the program allows for different objectives and scenarios to be easily experimented with, as the understanding of the dangers of contrail cirrus grows.

8.3 Requirement Evaluation

To confirm the success of this project, the initial requirements will be examined to ensure that all necessary requirements from Section 3.2 were met, with surrounding discussion around their challenges.

8.3.1 Weather Model

The weather model uses resampled data from the ECMWF ERA5 dataset [7], that is subsequently interpolated to a flight path upon request. The weather data is provided at a suitable resolution, as specified in Section 3.3.1. Therefore, the weather model meets all initial requirements.

8.3.2 Contrail Model

In order to reduce complexity, the contrail model uses an implementation of CoCiP from the *pycontrails* [34] library. Likewise, the 4D contrail field is downloaded and requested from the *pycontrails* API, which they granted access to for the purposes of this project.

8.3.3 Fuel Flow Model

The fuel flow model, uses the *OpenAP* [38] library for estimations of fuel flow data for different aircraft. These are then passed into the aircraft performance model (APM) to calculate total fuel consumption. Moreover, *OpenAP* provides emission estimations, using the fuel flow estimations. Therefore, all requirements of the fuel flow model were met.

8.3.4 Flight Path Construction

All requirements surrounding the construction of flight paths were met. A suitable grid of waypoints is constructed, between two specified points (although the option to specify these points is not available to the user). Moreover, the APM calculates flight characteristics for every single point along the path, taking into account information provided by the weather model for values such as ground speed and heading, and the fuel flow model for aircraft mass along the flight path. A separate consecutive points algorithm is used to enforce constraints on both the altitude and heading of the plane, as part of the routing graph. The implementation also provides a deep interface, meaning it is easy for external classes to calculate flight characteristics on a wide variety of flight path types.

8.3.5 Ant Colony Optimisation Algorithm

The implementation of m-ACO proved to be effective at considering several different objectives, and it provided flight trajectories that were better on all three objectives that were investigated when compared against two reference flights. Generated flight paths do sit within the aircraft's flight envelope though, as enforced by the APM. Whilst the algorithm does complete within under an hour, as a strict requirement, the goal to reduce this further was not met with total computation taking 30-40 minutes on a 16" 2019 MacBook Pro. Further efforts to improve on this could include experimentation with a hybrid parallel and multi-objective ant colony algorithm, similar to the one proposed by Mansour et al. [5].

Chapter 9

Conclusion and Future Work

9.1 Conclusion

This project has shown a novel use of the m-ACO algorithm as an effective evolutionary algorithm in the optimisation of aircraft flight paths, especially in the case of reducing the formation of contrail cirrus. Moreover, due to the multi-objective nature of the algorithm, this benefit comes without the trade-off of increased flight duration, or extra CO_2 emissions. It was found to be effective compared to both a randomly generated path, and a real transatlantic flight across all measured objectives.

9.2 Future Extensions

Whilst the project has been successful in its initial aims, there are plenty of room for improvements in future versions of this software that should be considered and discussed.

Perhaps the easiest to implement would be additional optimisation objectives. The system design allows extra objectives to be considered as part of the optimisation process, simply by declaring a new class for the objective. Therefore, the ACO algorithm could be extended to consider extra factors such as different emission types (NO_x or H_2O) or avoidance zones (such as over active war zones) when constructing optimised flight paths.

Another important addition that should be considered, is the optimisation of a fleet of aircraft rather than a single flight. Seemingly inefficient aircraft flight paths may be caused by factors such as congested airways. An algorithm that considers optimising several aircraft at once should theoretically then be able to provide greater optimisations overall than by

attempting to optimise individual flight paths. This would require major modifications to the program, and likely a different evolutionary algorithm. However, it is an interesting problem to consider for the future, and one that would potentially be more applicable to a real-world scenario.

Also, the optimisation process currently only considers the cruise phase of the flight, yet there is room for optimisation in both the climb and descent phases. This would require significant modification to implement, due to the differing requirements and flight parameters for each flight phase, but it is a conceivable addition that would allow for further optimisation. An approach similar to that by Mendoza et al. [9] could be taken to consider these extra flight phases. Although, in the context of specifically contrail formation this is less relevant, as contrails largely form during the cruise phase.

Currently, the program considers a static forecast of the weather, however this is not realistic for scenarios where you would want to deploy a program like the one described. A future extension of the program could consider a stream of weather forecasts, similar to what you would receive in a real-world scenario where the forecasts are being refined as you approach closer to departure. The estimation of the optimal flight path could then be continuously updated to match the forecast stream. It is likely that this addition would require little modification to the program too, as the weather data is already currently flexible, and so by running further iterations on a previously optimised path with the new weather forecast would likely incrementally adjust the path to match the new forecast. Although, this would need to be confirmed in a future iteration of the project.

As previously discussed though, the reduction of contrail cirrus is an active and ongoing area of research, meaning it is likely at least some of these future extensions will be explored, or are currently being explored.

References

[1] Kyoto protocol to the united nations framework convention on climate change. United Nations Framework Convention on Climate Change, 1997.

[2] World geodetic system — 1984 (wgs-84) manual. *International Civil Aviation Organization*, 2002.

[3] H Appleman. The formation of exhaust condensation trails by jet aircraft. *Bull. Amer. Meteor. Soc.*, 1953.

[4] Denis Avila, Lance Sherry, and Terry Thompson. Reducing global warming by airline contrail avoidance: A case study of annual benefits for the contiguous united states. *Transportation Research Interdisciplinary Perspectives*, 2:100033, 2019.

[5] Imen Ben Mansour, Ines Alaya, and Moncef Tagina. A new parallel hybrid multiobjective ant colony algorithm based on openmp. In *17th International Conference on Applied Computing*, Manouba, Tunisia, 2020. ENSI-COSMOS, University of Manouba; Esprit School Of Engineering.

[6] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[7] Copernicus Climate Change Service (C3S). ERA5 hourly data on single levels from 1940 to present. `10.24381/cds.adbb2d47`, 2023.

[8] Bogdan D Dancila and Ruxandra M Botez. Geographical area selection and construction of a corresponding routing grid used for in-flight management system flight trajectory optimization. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 231(5):809–822, 2017.

[9] R.I. Dancila and R.M. Botez. New flight trajectory optimisation method using genetic algorithms. *The Aeronautical Journal*, 125(1286):618–671, 2021.

[10] Design Specialist Group. Contrails and Contrail Management. *Royal Aeronautical Society*, 2023.

[11] Minnis P. Costulis P.K. Duda, D.P. and R Palikonda. Conus contrail frequency estimated from ruc and flight track data. 2003.

[12] European Environment Agency. *European Environment Agency CORINAIR Manual*, 2001.

[13] FlightAware. Flightaware. `https://www.flightaware.com/`. Flight tracking data and information.

[14] Alessandro Gardi, Roberto Sabatini, and Subramanian Ramasamy. Multi-objective optimisation of aircraft flight trajectories in the atm and avionics context. *Progress in Aerospace Sciences*, 83:1–36, 2016.

[15] John Green. Mitigating the climate impact of non-co_2 – aviation's low-hanging fruit. Report from the RAeS 'Mitigating the Climate Impact of Non-CO_2 – Aviation's Low-Hanging Fruit' Virtual Conference, March 2021.

[16] Aric Hagberg, Pieter Swart, and Daniel Chult. Exploring network structure, dynamics, and function using networkx. 01 2008.

[17] Khaled Ghedira" Ines Alaya, Christine Solnon. Ant colony optimization for multi-objective optimization problems. *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2007.

[18] International Energy Agency. Aviation - energy systems. `https://www.iea.org/energy-system/transport/aviation`. Retrieved from IEA website.

[19] S. Kahn Ribeiro P. Newman S. Dhar O.E. Diemuodeke T. Kajino D.S. Lee S.B. Nugroho X. Ou A. Hammer Strømman J. Whitehead Jaramillo, P. Climate change 2022: Mitigation of climate change. 2022.

[20] Charles F. F. Karney. Algorithms for geodesics. *Journal of Geodesy*, 87(1):43–55, 2013.

[21] David S. Lee, Myles R. Allen, Nicholas Cumpsty, Bethan Owen, Keith P. Shine, and Agnieszka Skowron. Uncertainties in mitigating aviation non-co2 emissions for climate and air quality using hydrocarbon fuels. *Environ. Sci.: Atmos.*, 3:1693–1740, 2023.

[22] Yixiang Lim, Alessandro Gardi, and Roberto Sabatini. Modelling and evaluation of aircraft contrails for 4-dimensional trajectory optimisation. *SAE International Journal of Aerospace*, V124, 09 2015.

[23] Yixiang Lim, Alessandro Gardi, and Roberto Sabatini. Optimal aircraft trajectories to minimize the radiative impact of contrails and co2. *Energy Procedia*, 110, 04 2017.

[24] Jarlath Molloy, Roger Teoh, Seán Harty, George Koudis, Ulrich Schumann, Ian Poll, and Marc E. J. Stettler. Design principles for a contrail-minimizing trial in the north atlantic. *Aerospace*, 9(7), 2022.

[25] Alejandro Murrieta Mendoza, Audric Bunel, and Ruxandra Botez. Aircraft lateral flight optimization using artificial bees colony. 11 2015.

[26] Alejandro Murrieta Mendoza, Antoine Hamy, and Ruxandra Botez. Four- and three-dimensional aircraft reference trajectory optimization inspired by ant colony optimization. *Journal of Aerospace Information Systems*, 14:1–20, 10 2017.

[27] Alejandro Murrieta Mendoza, Hugo Ruiz, Sonya Kessaci, and Ruxandra Botez. 3d reference trajectory optimization using particle swarm optimization. 06 2017.

[28] Angela Nuic, Damir Poles, and Vincent Mouillet. Bada: An advanced aircraft performance model for present and future atm systems. *International Journal of Adaptive Control and Signal Processing*, 2010. Published online in Wiley InterScience (www.interscience.wiley.com).

[29] PROJ contributors. *PROJ coordinate transformation software library*. Open Source Geospatial Foundation, 2023.

[30] Yaohong Qu, Yintao Zhang, and Youmin Zhang. Optimal flight path planning for uavs in 3-d threat environment. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 149–155, 2014.

[31] J Rosenow and H Fricke. Luchkova et al. minimizing contrail formation by rerouting around dynamic ice-supersaturated regions. *Aeron Aero Open Access J*, 2(3):105–111, 2018.

[32] U. Schumann. A contrail cirrus prediction model. *Geoscientific Model Development*, 5(3):543–580, 2012.

[33] Ulrich Schumann and Andrew J. Heymsfield. On the life cycle of individual contrails and contrail cirrus. *Meteorological Monographs*, 58:3.1 – 3.24, 2017.

[34] Marc Shapiro, Zeb Engberg, Roger Teoh, Marc Stettler, and Tom Dean. pycontrails: Python library for modeling aviation climate impacts, November 2023.

[35] Joel Silverberg. Napier's rules of circular parts. Presented at the University of British Columbia, Vancouver, British Columbia, Canada, June 2008. Special Session on Trigonometry and Its Applications. Canadian Society for the History and Philosophy of Mathematics.

[36] Thomas Stützle and Marco Dorigo. Ant colony optimization. 01 2004.

[37] Thomas Stützle and Holger Hoos. Max-min ant system. 16, 11 1999.

[38] Junzi Sun, Jacco M Hoekstra, and Joost Ellerbroek. OpenAP: An open-source aircraft performance model for air transportation studies and simulations. *Aerospace*, 7(8):104, 2020.

[39] R. Teoh, U. Schumann, E. Gryspeerdt, M. Shapiro, J. Molloy, G. Koudis, C. Voigt, and M. E. J. Stettler. Aviation contrail climate effects in the north atlantic from 2016 to 2021. *Atmospheric Chemistry and Physics*, 22(16):10919–10935, 2022.

[40] Roger Teoh, Ulrich Schumann, Arnab Majumdar, and Marc E. J. Stettler. Mitigating the climate forcing of aircraft contrails by small-scale diversions and technology adoption. *Environmental Science & Technology*, 54(5):2941–2950, 03 2020.

[41] Etienne Terrenoire, Didier A. Hauglustaine, Yann Cohen, Anne Cozic, Richard Valorso, Franck Lefèvre, and Sigrun Matthes. Impact of present and future aircraft NOx and aerosol emissions on atmospheric composition and associated direct radiative forcing of climate. *Atmospheric Chemistry and Physics*, 22:11987–12023, 2022. Research article.

[42] United Nations. Paris agreement. `https://unfccc.int/sites/default/files/english_paris_agreement.pdf`, 2015. Retrieved from UNFCCC website.

[43] J. Haigh D. Hauglustaine J. Haywood G. Myhre T. Nakajima G.Y. Shi S. Solomon V. Ramaswamy, O. Boucher. Radiative forcing of climate change. *IPCC*.

[44] Florent Vergnes, Judicaël Bedouet, Xavier Olive, and Junzi Sun. Environmental impact optimisation of flight plans in a fixed and free route network. 06 2022.

[45] Bing Wang, Hemant Kumar Singh, and Tapabrata Ray. Adjusting normalization bounds to improve hypervolume based search for expensive multi-objective optimization. *Complex & Intelligent Systems*, 9(2):1193–1209, 2023.

[46] Tong Wei, Manzhen Duan, Bin Dong, Yinfeng Li, and Shenghao Fu. *Rerouting Path Planning Based on MAKLINK Diagram and MS-Genetic Algorithm*, pages 251–260. 02 2021.

[47] Wikipedia. Great-circle navigation — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Great-circle\%20navigation&oldid=1182102692`, 2023. [Online; accessed 23-November-2023].

[48] Dabin Xue, Kam K.H. Ng, and Li-Ta Hsu. Multi-objective flight altitude decision considering contrails, fuel consumption and flight time. *Sustainability*, 12:6253, 08 2020.

[49] Saúl Zapotecas-Martínez, Antonio López-Jaimes, and Abel García-Nájera. Libea: A lebesgue indicator-based evolutionary algorithm for multi-objective optimization. *Swarm and Evolutionary Computation*, 44:404–419, 2019.

[50] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.