Onboard Flight Dynamics Route Optimization*

João Portugal[1]

Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal

Abstract. Nowadays the flight management systems (FMS) of an airplane can fly it autonomously from takeoff to landing with little intervention from the pilots assuming there are no anomalous events. However, some events require a diversion and subsequent route replanning. This replan is non-trivial, subject to a restricted criteria and takes attention away from the pilots and from air traffic controllers, attention that could be spent monitoring other flight systems. Route planning involves finding a new efficient route, communicate it to ATC, receive approval and reprogram the flight computer to follow the new path. Our work presents a module capable of finding a feasible flight path, removing the need of pilot intervention in this area lightening up their work as well as the air controller's work. We model the problem as a knapsack model and we then utilize multi-objective problem solving methods to efficiently recalculate the route.

Keywords: multi-objective problem · route planning · FMS · knapsack model

* Supported by Instituto Superior Técnico.

Table of Contents

1 Introduction

1.1 Motivation

Aviation is growing every year and is expected to double over the next 20 years [11] and more and more challenges surrounding airplanes such as safety, eco-friendliness and increasing air traffic become more and more discussed.

Airplanes are capable of flying autonomously from take-off to landing when considering standard normal operations but when trouble arises, pilot intervention is often needed [1]. Some of these issues require the pilots to adjust the route, sometimes the adjustment ending in a completely new destination.

The issues can be categorized into two different types: internal and external. Internal issues are problems the airplane might be having internally such as engine failure or a medical emergency. External issues are problems outside of the airplane such as weather storms or heavy traffic that might pose danger to the aircraft.

Some of the possible routes are already predetermined by the pilots prior to departure but sometimes a deviation is needed and when that happens the new route found may not be optimal. When considering what is optimal, there is a plethora of possible restrictions and criteria to follow but aircraft safety, fuel efficiency are usually the ones that matter the most.

Flight planning is the process of producing a flight plan that describes a trajectory to be taken from a start point to an end point. It involves calculating how much fuel is needed, the route that takes us to the arrival point safely while complying with ATC regulations and safety rules. Flight planning depends on a lot of factors. Depends on the distance, the weather conditions and the aircraft used. Companies are always required to take a surplus of fuel for safety reasons. This planning is not trivial and is never a one-time process. Varying weather conditions such as tail wind might increase or decrease the fuel needed. Weather storms might block some waypoints requiring the plane to take one another. It is important to note that an airplane has restrictions on its own movement, subject to its own flight dynamic restrictions, meaning that for example it cat not change altitude instantaneously or change its course suddenly without affecting either passenger comfort or even the safety of the aircraft itself. This also has to be taken into account in flight planning.

There are already several flight planners able of generating routes, however when the aircraft is already airborne, it becomes dependent on the pilots manually adjusting the course of the route through ATC advice. And this is something that can be theoretically fixed with an efficient on-board flight-planner that can receive input to changing conditions that might require route adjustments and automatically adjust it without taking away attention from the pilots, letting them monitor flight systems more closely.

We propose a flight-planner that can efficiently recalculate a route in the presence of adversarial conditions and automatically adjust the trajectory of the route while maintaining optimality, safety and being compliant with the restrictions imposed.

The rest of this chapter defines the objectives of this thesis, the hypothesis raised as a solution and the outline of the thesis proposal.

1.2 Objectives

At the present day, routes are generated before the airplane takes off and if there is a problem that surges when the aircraft is already airborne, like a weather storm, the pilots and/or the ATC have to manually re-plan the route to avoid that problem [1][11][24].This maybe just a change of one or two waypoints or a much more complex process that can even end in finding a new destination. A waypoint is an intermediate point in a route at which course is changed leading to a new waypoint. As seen in Fig. 1, KCPS and KMDW are respectively the departure and arrival points while CSX, TEHWY, SPI, ...,etc., are waypoints.

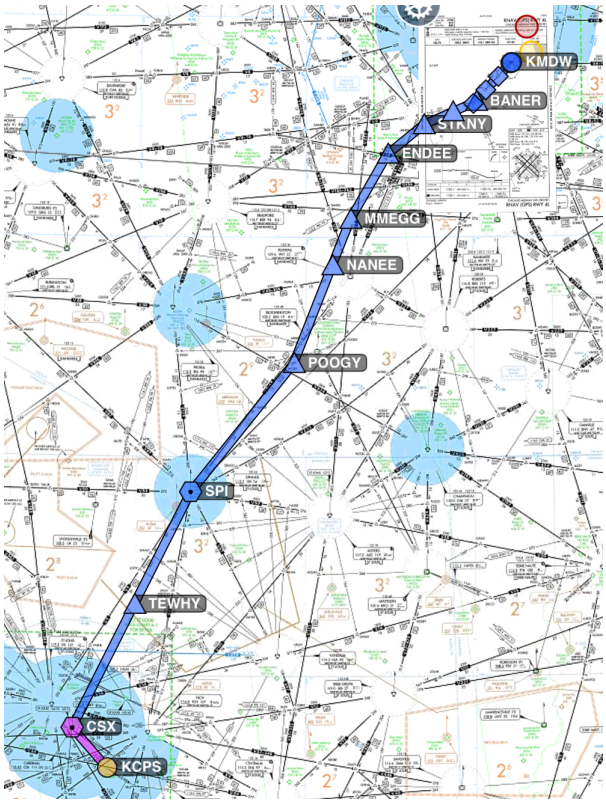

Fig. 1. Waypoints example.

This thesis focus mainly on how to efficiently calculate and adapt routes while the aircraft is already airborne by utilizing the current position of the aircraft and avoid all the obstacles that arise while full-filling the requirements set. Our solution intends to prove that an on-board optimizer is efficient and can be used as a replacement to manual recalculations of the route.

There are already very efficient and fast algorithms such as the randomized potential field [8], the random walk-planner [9] and the rapidly exploring random tree, however this solutions are based on random sampling leading to potential non-optimal solutions when subject to a pre-specified criterion.

We first prove that the problem can be represented as a multi-objective problem and then utilize well-known solving methods of this type of problem to calculate the optimized route.

In this work, we do the following path in order to obtain the optimal route:

1. Find the optimal geometrical path (trajectory) between 2 points in a 3D space with possible obstacles.
2. Find the optimal trajectory between 2 points in a 3D space with possible obstacles while it being feasible. This means adding restrictions that the plane is subject to such as speed or the rate-of-climb of the aircraft.

While there are numerous situations where route recalculation might be needed, we restrict this type of problems to three situations:

1. Finding the optimal route considering weather problems.
2. Finding the optimal route considering there is a situation that requires the plane to land as soon as possible.
3. Considering both option (1) and (2) and adding ETOPS [1] restrictions to it. ETOPS (Extended-range Twin-engine Operational Performance Standards is a set of standards that permits twin-engine aircraft to fly routes which at some point are more than 60 minutes of flying time from the closest airport for an eventual emergency landing.

These situations are solvable in if we can solve the final part of the gathering of the optimal route in which all restrictions are possible.

To show that our solution is optimal we use the F16 as a plane model, a military fighter jet widely used, and we use JSBSim, an open source flight dynamics model tool that can generate flight routes and we can compare those flight routes to our own generated ones.

1.3 Outline

The remainder of the proposal is composed by 5 chapters.

In the next chapter we present the State of the Art where we describe the main concepts related to pathfinding, route optimization and the knapsack model. We also describe existing solutions to route planning.

[1] `http://www.boeing.com/commercial/aeromagazine/articles/qtr_2_07/article_02_3.html`

In Chapter 3 we propose a solution and its architecture with reasoning as to why it will be a good and efficient solution.

In Chapter 4 we propose an evaluation methodology to be taken to try and prove that our solution is optimal.

In Chapter 5 we make a calendarization of the work to be taken where we make rough estimates on how long it will take at each stage progress.

In the last chapter we make a conclusion of the proposal with a general discussion of the challenges that lie ahead.

2 State of the Art

2.1 Pathfinding

Introduction

Pathfinding has been a problem studied throughout the years. It can be considered as the process of finding the shortest path between two nodes with or without obstacles in between. Djikstra's algorithm invented in 1956 [10] is the simplest form of pathfinding in a weighted graph. Since then, a multitude of pathfinding algorithms has been invented and improved on: A* [22] is an improvement on Dijkstra's algorithm where instead of just using the real costs per node, the search is now conduted based on heuristics. Ford-Fulkerson [14] is an algorithm that computes the maximum flow of a flow network, where we can find optimal paths in arcs that can have at max x flow. All this research has led to each situation requiring a specific algorithm and this algorithm could be adapted into different situations. For example, Bellman-Ford [14] can be used with or without a node queue and could be parallelized if some conditions are verified.

Pathfinding in transportation

In transportation, pathfinding has always been a problem. Everyone wants to get to their destination in the best way possible, but sometimes what is best is subjective. While some prefer to pay tolls to save time, others would rather go through toll-free roads and save money instead of time.

In 1931, the Zermelo's Navigation Problem [26] was the first posed optimal control problem. The problem was to find an optimal path for a boat stuck in water with water currents and wind. If we do not consider the presence of these obstacles, the optimal path is a straight line from start to finish relating to the shortest path. However, when considering these obstacles, the optimal path is usually not a straight line. The same analogy can be used in aviation, obstacles can represent weather conditions or restricted airspace for example, leading to a path that is not a direct line from starting point to end point. Route planning has always been target of research and so there an uncountable number of thesis and works that describe multiple ways of finding the optimal path. We will describe some that pertains to our work the most.

The main goal of guidance applied to navigation is to provide a reference velocity V, Path Angle γ and Heading ψ to enable the aircraft to follow waypoints $P_0, P_1, ..., P_N$. However, there are multiple ways of finding these parameters. The FMS is already programmed so that if provided a waypoint, it can automatically give a path angle and heading into that location. The 3D path planning problem has been shown to be NP hard [6] but many solutions have been already presented.

In 2003 Myungsoo Jun et al. proposed a method of path planning using a map of thread probabilities made using surveillance data and from there create the optimal route [19]. It starts by determining occupancy values based on sensor readings and then applying the conditional probability of occupancy using Bayes' rule. Then, they generate a digraph from the probability map in order to convert the model to a shortest path problem. To find the optimal path, the authors chose to use Bellman-Ford algorithm [4] because of the flexibility it provides, since changes in the probability map are probable and would change the route and so they could update link lengths without having to stop and restart the algorithm. They also were able to make the Bellman-Ford asynchronous and distributed further improving efficiency. The results showed that the algorithm could generate paths although sometimes it was the safest path and not the optimal one. The main advantage of this solution is that it is very fast in computing a path and is compatible with distributed computation. However, things like frequent acceleration and deacceleration were not considered, which would need to be improved on, since this changes fuel consumption potentially leading to a sub-optimal path in the end.

In 2006 Igor Alonso-Portillo et al. proposed an adaptive trajectory planner capable of adjusting its world model and re-computing feasible flight trajectories in response to adversarial changes [1]. The module proposed was to be included into the FMS and it could transmit information that is not currently being transmitted such as engine failure, control surface jams, anything that could cause flight dynamics to vary and therefore modify the aircraft perfomance. The proposed module was not to be seen as a replacement of the FMS but as improvement to the system robustness. The module when detecting a failure updates the flight dynamic model accordingly, generates a footprint and starts a search for a suitable landing site. This landing site is chosen from an existing database of airports that contains information such as airport location, runway specifications amongst other relevant information. The footprint generated allows to find reachable airports. After this, the module performs a constraint analysis to select minimally safe airports and the constraints are repeatedly relaxed until at least one solution is found. Then an utility function is applied based on the airport characteristics to find the most suitable airport for an emergency landing. Finally, a trajectory to that airport is created utilizing existing tools on the FMS. Results showed that the module provided robustness to the FMS to different failure modes although more work on the generation of the trajectory was needed including taking into account wind and current weather.

In 2009 David Šišlák et al. proposed a variation of the A*, the AA* (Accelerated A*) [24]. The main difference is the reduction of the state space when calculating the trajectory planning. While the precision in finding optimal paths stays around the same as in the A* star algorithm, its memory space is reduced up to 1400 times lower. AA* introduces adaptive sampling: during the expansion, child states are generated by applying vehicle elementary motion actions using adaptive parametrization. These actions are defined by a model of non-holonomic airplane movement dynamics. The parametrization varies based on the distance to the nearest obstacle. The algorithm then finds a path to the airplane which is bounded by a sphere. The results showed that the solution proposed was effective and efficient, providing an acceleration of up to 1400 times in the planning in both the case on where a path existed and in the case of no available path existing, significantly reducing the number of visited states in the search.

Alireza Babaei et al. in 2010 proposed an efficient algorithm for trajectory planning in autonomous unmanned aerial vehicles [2]. A Dubins path is a path that is the shortest path between two points satisfying curvature constraints. The authors propose a Dubins path based algorithm for UAV's in a 3D space considering wind-free space that exploits analytical and differential geometry. The application of Dubins path to 3D space is presented in this paper. The trajectory planning proposed as the form of a closed-loop guidance form that generates commands in terms of waypoint configuration and the maximum curvature allowed. The results showed that the solution was effective and was suggested for dynamic environments. However, further research would be needed, since wind is not taken into account in this work.

In 4D trajectories, waypoints are defined as 3D waypoints but additionally having arrival time at each waypoint specified.

Bousson et al. in 2010 [5] proposed a 4D approach where they transformed the problem in a nonlinear programming problem using pseudospectral integration and Chebyshev polynomials. The method proposed had the advantage of having the expected arrival time calculated at each waypoint in addition to the desired position. They first formulated an optimal control problem and then transformed it into a nonlinear programming problem by parametrizing the state and control. This parametrization is important to the accuracy of the solution and in this work has been done via pseudospectral methods and then compared to collocation methods, the most known techniques at the time. Results showed that this approach achieved appropriate solutions for the benchmarks set, which were time, path angle, heading, latitude, longitude, height and control, having the collocation methods failed to find solutions in one example while the pseudospectral methods was able to find for both examples provided. Even not considering this, the solutions generated were smoother than the ones presented by the Collocation method.

In 2013, Nourelhouda Dougui et al. proposed the light propagation algorithm, an algorithm capable of generating sets of conflict-free 4D trajectories [11]. The algorithm is based on Fermat's principle of least action: *The path of a light ray*

connecting two points is the one for which the time of transit, not the length, is a minimum.. It finds an optimal path by computing smooth geodesic trajectories in environments with obstacles. Light tends to travel in low index areas where light rays are slowed down. Having this in mind, the algorithm programs obstacles as high-index areas, thus mimicking the light propagation behaviour. They were able to successfully apply on three different air traffic management problems. However, the authors noted that the algorithm still needed and could be optimized by computing the cluster resolutions in parallel and that the algorithm needs further improvements in more general restrictions, since it only accounted for temporal congestion and moving weather.

Arianit Islami et al. proposed in 2016 a large scale 4D trajectory planning system [17] in which they aimed to minimize the global interaction to reduce workload for both pilots and air traffic controllers. With this work, the authors intended to propose a global deconfliction module that could predict and plan trajectories such a way that two airplanes would never be in conflict even in high density traffic. They proposed a mathematical formulation to achieve a trajectory planning in which they allocate alternative departure times and alternative horizontal and vertical paths. This mathematical formulation was split in three parts. The first one characterizes the uncertainty of aircraft positions and arrival times based on two different models. Then, they define interact-ion between trajectories. Finally, they present a mathematical formulation for the interaction minimization problem. After this section, they present a hybrid metaheuristics approach adapted to handle an air-traffic assignment problem at a continent scale. Since this model is based on uncertainty there is a trade off between the robustness of the solution and the trajectory modification costs. The authors proposed that the user could consider lower uncertainty levels and iteratively solve the remaining interactions in another phases. However, this is computationally expensive since at each step we would need to recalculate if a space was clear or not and if so recalculate all the possible trajectories.

In 2017 Arno Fallast et al. proposed an automated trajectory generator for an emergency landing procedure for a CS23 aircraft [12]. The finding of an emergency landing includes selecting possible landing sites and from there create feasible trajectories. This procedure is separated in 3 processes: the module responsible for the route generation to the intended landing sites, the module that searches for available landing sites and the last module which is responsible for combining both the search for available landing sites and the route generation scoring each trajectory. The results show that the algorithm is capable of finding an optimal solution if there exists one, this solution being close to the unknown true optimum. The authors note that the work is still on-going and that further testing is necessary if using this generator.

In 2018, Russel Paielli proposed a standard Trajectory Specification Language (TSL) [21] where at any given time, they could specify an aircraft trajectory with tolerances that the aircraft would always be constrained to a precisely defined bounding space. The main benefits of using Trajectory Specification is in case of a system outage, risks are minimized since even if ATC fails, safety

must be guaranteed and so, if a explicit bounding deviation from the trajectory assigned is computed, then a safe separation can be assured within a time threshold previously defined known as the conflict-free time horizon. TSL is proposed as a way to achieve these specifications and to communicate them via air/ground datalink. From air to ground, the language would be used as a trajectory request service while from ground to air, the language would be used as a trajectory assignment.

2.2 Knapsack Problem

Introduction

The knapsack problem is a combinatorial optimization problem and it talks about the common problem of packing the most valuable things without overloading the luggage. Formally, it seeks to select from a finite set of items, the subset that maximizes the linear function of the items chosen, subject to constraints [7]. It can be described as:

$$\begin{aligned} max f_1(y) &= \sum_{j=1}^{n} v_j^1 y_j \\ &\vdots \\ max f_r(y) &= \sum_{j=1}^{n} v_j^r y_j \\ s.t. \sum_{j=1}^{n} w_j y_j &\leq W \end{aligned}$$

where:

- n is the number of items
- r is the number of criteria
- v_j^k is the kth profit of the jth item, for $k = 1, ..., r$
- w_j is the weight of item j
- W is the capacity of the knapsack

We can define dominance using the following definition [18]:

Let z' and z" $\in \mathbb{R}^n$ denote two criterion vectors. z' dominates z" if z' $\geq$ z" and z' $\neq$ z". z' strongly dominates z" if and only if z' $>$ z".

Non-dominance can be defined as:

Let $\overline{z} \in \mathbb{R}^n$. z dominates $\overline{z}$ if there does not exist $z \in Z$ such that $z \geq \overline{z}$ and $z \neq \overline{z}$. Otherwise, $\overline{z}$ is a dominated vector.

A knapsack model can modeled into a pathfinding problem [7] thus an optimal trajectory can be found using this model. Indeed, we can choose as objectives

what we want to maximize or minimize in our set, such as fuel costs or travel time and set the restrictions as to things like ETOPS rules, ATC rules, passenger comfort, ..., etc.

Solutions

In 2003, M. Eugenia Captivo et al. proposed a method in *Solving bi-criteria 0–1 knapsack problems using a labeling algorithm* [7] where we can formulate the knapsack problem as a shortest path problem [20] by transforming the multiple criteria knapsack problem into a multiple criteria shortest path problem over an acyclic network and then proceed to label it. It first converts a knapsack model to an acyclic network model.

To convert the model to an acyclic network we first need to determine the set of nodes and the set of arcs and respective arc costs. The set of nodes can be found by using a layer technique. The first layer has only the starting node, each progressive intermediate layer can contain several nodes and the last layer only has one node. Layer j can be directly obtained from layer $j-1$ where each layer has at most $W+1$ nodes.

The starting node s always has 2 outgoing arcs, one with cost 0 and the other with cost $-v_1$. The first arc represents the decision of not including item 1 in the knapsack while the second one represents the decision to include it. After that, in each layer, each node j^a where $a = 0, ..., W$ has at most 2 arcs. The first arc has cost 0 meaning that item $j+1$ is not included in the knapsack. The second arc has cost $-v_j p1$ if $a + w_j + 1 \leq W$ meaning that, node $j+1$ is included in the knapsack if and only if the knapsack has capacity to contain this item. All nodes from the final intermediate layer are connected to the final node t.

The pseudo-algorithm to convert a knapsack model to a shortest path problem model is depicted in Figure 2.

The network generated by this algorithm has no cycles, every feasible solution of the knapsack problem has a corresponding path in the network generated from starting node s to end node t and the shortest path from s to t represents the optimal solution for the knapsack model and the value is the negative cost of the shortest path in the network.

This transformation is only for a single criterion but the generalization is very easy to do. Instead of just assigning one cost per arc, we assign a vector of costs per arc, each cost corresponding to one different criterion.

After converting it, to label it, they order it lexicographically but adding an additional property that other labeling algorithms do not have: the values concerning the first criterion are placed in non-decreasing order, while the values of the second one are placed in non-increasing order. So, to see if a new given label is dominated or not, it is only necessary to compare this new label with the last non-dominated label determined. For more than two criteria the new label must be compared with all the labels already determined. Figure 3 shows the labeling algorithm proposed by the authors where T and V are two lists of superscripts related to the nodes in layer j and $j+1$ and $S(j^a)$ is the set of non-dominated labels concerning the set of paths from s to j^a.

Building a network shortest path model.
{ Converting a knapsack model into a shortest path problem. }
(1) begin
(2) Set $\mathcal{Q} \leftarrow \{s\}$, $T \leftarrow \{0, w_1\}$ and $V \leftarrow \{\}$;
(3) $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{1^0, 1^{w_1}\}$;
(4) Set $\mathcal{A} \leftarrow \{(s, 1^0), (s, 1^{w_1})\}$;
(5) $c(s, 1^0) \leftarrow 0$ and $c(s, 1^{w_1}) \leftarrow -v_1$;
(6) **for** $(j = 1)$ **to** $(n - 1)$ **do**
(7) **begin**
(8) **while** $(T \neq \{\})$ **do**
(9) **begin**
(10) Let a be the first element in T and remove it from T;
(11) **if** $(a \notin V)$ **then**
(12) **begin**
(13) $V \leftarrow V \cup \{a\}$;
(14) $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j+1)^a\}$;
(15) **end**
(16) $\mathcal{A} \leftarrow \mathcal{A} \cup \{(j^a, (j+1)^a)\}$;
(17) $c(j^a, (j+1)^a) \leftarrow 0$;
(18) **if** $(a + w_{j+1} \leq W)$ **then**
(19) **begin**
(20) **if** $(a + w_{j+1} \notin V)$ **then**
(21) **begin**
(22) $V \leftarrow V \cup \{a + w_{j+1}\}$;
(23) $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(j+1)^{a+w_{j+1}}\}$;
(24) **end**
(25) $\mathcal{A} \leftarrow \mathcal{A} \cup \{(j^a, (j+1)^{a+w_{j+1}})\}$;
(26) $c(j^a, (j+1)^{a+w_{j+1}}) \leftarrow -v_{j+1}$;
(27) **end**
(28) **end**
(29) $T \leftarrow V$ and $V \leftarrow \{\}$;
(30) **end**
(31) $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{t\}$;
(32) **while** $(T \neq \{\})$ **do**
(33) **begin**
(34) Let a be the first element in T and remove it from T;
(35) $\mathcal{A} \leftarrow \mathcal{A} \cup \{(n^a, t)\}$;
(36) $c(n^a, t) \leftarrow 0$;
(37) **end**
(38) end

Fig. 2. From knapsack model to shortest path problem. Taken from Captivo et al.

In 2009 Cristina Bazgan et al. proposed an algorithm to efficiently solve the 0-1 multi objective knapsack problem [3]. The main idea of the algorithm was to disregard partial solutions that would lead to new non-dominated criterion vectors by using several complementary dominance relations. It was claimed to be the most efficient algorithm existing with up to 4000 items in less than 2hrs in the bi-objective case comparing the results obtained with Captivo et al. [7], the most efficient method at the time.

In 2013, J. Figueira et al. in *Algorithmic improvements on dynamic programming for the bi-objective 0,1 knapsack problem* [13] proposed an optimization to the at the time state-of-art algorithm which was the one proposed by Bazgan [3]. The optimizatin was based on applying several dominance relations to discard elements from the solution pool. It proposed 3 techniques to do this:

Computing efficient paths in the network.
{ A new labeling algorithm to determine efficient paths. }
(1) **begin**
(2) Set $S(1^0) \leftarrow \{(0,\ldots,0)\}$ and $S(1^{w_1}) \leftarrow \{(-v_1^1,\ldots,-v_1^r)\}$;
(3) Set $T \leftarrow \{0, w_1\}$ and $V \leftarrow \{\}$;
(4) **for** $(j = 2)$ **to** n **do**
(5) **begin**
(6) **for** $(a = 0)$ **to** W **do**
(7) **begin**
(8) **if** $(a \in T)$ **then**
(9) **begin**
(10) $V \leftarrow V \cup \{a\}$;
(11) **if** $(a - w_j \in T)$ **then** { two incoming arcs are defined }
(12) $S(j^a) \leftarrow$ Set of ND labels of $\Big(S\big((j-1)^a\big) \cup$
(12a) $\{(-v_j^1,\ldots,-v_j^r)\} + S\big((j-1)^{a-w_j}\big)\Big)$;
(13) **else** { only the arc $\big((j-1)^a, j^a\big)$ is defined }
(14) $S(j^a) \leftarrow S\big((j-1)^a\big)$;
(15) **end**
(16) **else**
(17) **begin**
(18) **if** $(a - w_j \in T)$ **then** { only the arc $\big((j-1)^{a-w_j}, j^a\big)$ is
(18a) defined }
(19) **begin**
(20) $S(j^a) \leftarrow \{(-v_j^1,\ldots,-v_j^r)\} + S\big((j-1)^{a-w_j}\big)$;
(21) $V \leftarrow V \cup \{a\}$;
(22) **end**
(23) { **else** The node j^a does not exist }
(24) **end**
(25) **end**
(26) $T \leftarrow V$ and $V \leftarrow \{\}$;
(27) **end**
(28) $S(t) \leftarrow$ Set of ND labels of $\bigcup_{a=0}^{W} S(n^a)$;
(29) **end**

Fig. 3. Labeling Algorithm. Reproduced from Captivo et al.

1. Computing a small set of efficient solutions by solving a sequence of weighted scalarized bi-objective 0, 1 knapsack problem with dichotomic search.
2. Computing the nondominated extreme vectors of the relaxation of the problem solved by a bi-objective simplex algorithm and then using the *improvement* method [15] to restore feasibility.
3. Generating feasible extensions for each partial solution at a given stage k by adding its profit to the profits obtained through dichotomic search on the reduced bi-objective $0, 1$ knapsack problem with the remaining $n-k$ objects.

The experiments led using these techniques resulted in average of a 20% CPU time improvement and significant less memory used than in the original method. The set of instances as wide and the results showed improvement even in the most difficult problems in the literacy.

In multi-objective problems its very rare to reach only one optimal (or non-dominated) solution. Instead, we reach what is called a Pareto front, a set of

nondominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective.

There are techniques designed to pick one of these solutions. We will talk about two of the most known ones: TOPSIS [16] and AHP [23]

The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) was proposed by Yoon and Hwang in which the ideal solution would be the one to have the shortest distance to the ideal solution but also the farthest distance from the negative-ideal solution. TOPSIS assumes that each attribute in the decision matrix takes either monotonically increasing or monotonically decreasing utility. With this in mind, it is easy to locate the ideal solution, this is the one being the closest to the optimum but farthest from the non-optimum. For example, if a solution S_1 has the closest distance to the optimum O but also the closest distance to the non-optimum NO while a solution S_2 is close to O but the farthest away from NO then it gets really hard to justify the selection of S_1.

The Analytic Hierarchy Process (AHP) is a technique for organizing and analyzing complex decisions proposed by Thomas Saaty. Here, factors relevant to the decision are arranged in a descending hierarchy from an overall goal to criteria, sub-criteria and alternatives in successive levels.

When constructing the hierarchy is important to include enough relevant detail to the environment surrounding the problem, the correct identification of both the issues and the attributes that surround the problem and to identify the participants associated with the problem. The hierarchy serves two purposes: an overall view of the problem and the quantification of the magnitude of the issues; if two are on different levels then it is easier to compare them accurately.

In AHP a scale of measurement consists of three elements: a set of objects, a set of numbers and a mapping of the objects to the numbers. The numbers are measures, and while in standard scales measures could be standard units such as meters or radians, AHP generates relative ratio scales of measurement, these ratios being obtained through normalization. However, the normalization and composition of weights of alternative with respect to more than one criterion measure on the same standard scale it leads to nonsensical numbers since normalizing separate sets of numbers destroys the linear relationship between them. In order to be able to use AHP, the weights must be first composed and then normalized. After building these sets, we can get the priorities by adding benefits and subtracting costs of decisions, getting the optimal decision.

2.3 Pathfinding State of the Art survey

In 2013, Omar Souissi et al. did a survey and presented a state of the art in path planning in the field of automation, robotics and video-games [25].

Figure 4 shows an overall hierarchy of how pathfinding problems can be classified according to these authors.

Overall, the authors present a lot of different ideas to model a problem and algorithms that could be useful in modern day problems. However, in this paper the authors claim that there is a separation in the second level where one can

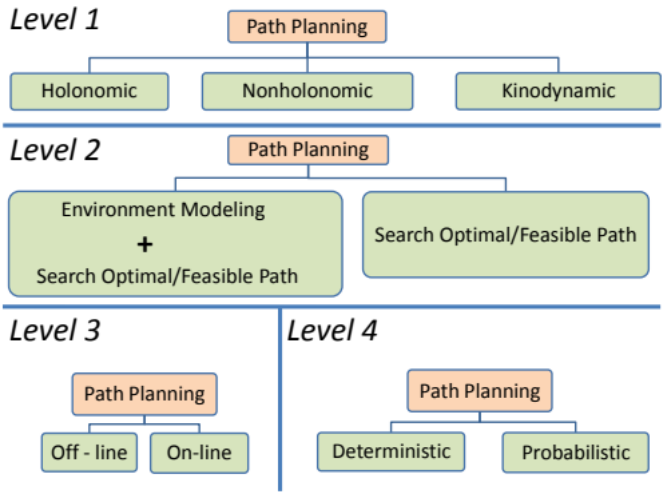

Fig. 4. Path planning levels. Reproduced from Souissi et al.

either have environment modeling and not so optimal solutions or disregard the environment modeling and get optimal or near optimal solutions.

2.4 OODA Automation

OODA (observe-orient-decide-act) is an information strategy concept developed by Colonel John Boyd for information warfare. It was created for military purposes, although nowadays is used for business and comercial strategies. According to Boyd, decision-making occurs within a cycle: firstly, make observations on the world around the system (the enemy). After that, an orientation would follow. This consisted on the formulation of a decision hypothesis. This decision would take things like cultural tradition and previous experience as basis for the hypothesis. After that, the decision making would take place and finally it would be execution of the decision.

Boyd argued that whoever completed the OODA cycle would win, since if we executed the decision before our enemies we would lengthen their decision making since they would have to process our movement.

Figure 5 shows an overall view of the OODA cycle.

In this cycle, feedback is the concept that decisions do not have to necessarily become actions. The optimal course may remain the same, even if some condition changed. The implicit guidance and control means that the process occurs in the background and so the course only changes when the decision is final and not before that.

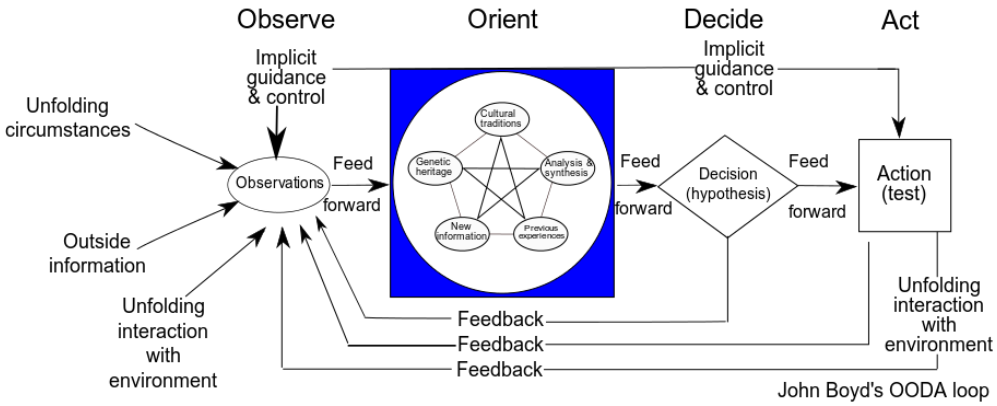

Fig. 5. OODA cycle

3 Solution Proposal

All the previous work, while achieving relevant results to their own studies, none presented a clear module that worked efficiently that could be used online in an autonomous non-tripulated vehicle considering a wide number of restrictions. While some proposed a module for an emergency landing [12], it would be limited to certain types of aircraft (CS23). Others, do not consider wind variations or another type of restrictions in their planning.

Our module receives as input the current trajectory in the form of: current position, the waypoints to be followed after and the destination point. These positions are represented in the form of *Latitude Longitude Altitude*. In addition to the trajectory, we receive a series of restrictions as well.

These restrictions are split in two big groups: Aircraft restrictions and route restrictions. Aircraft restrictions are constraints that the aircraft has such as speed, turns and fuel consumption. These restrictions are variable and depend on which aircraft model is being used. The route restrictions can be divided into two sets: dynamic and static. The static restrictions are set restrictions by the ATC known prior to departure to the pilots such as restricted airspace, also known as no-fly zones, or zones with limited speeds. Dynamic restrictions are restrictions that occur when the aircraft is already airborne. Weather storms, medical emergencies, all of these constitute dynamic restrictions.

As output our module returns the replanned route in the form of a start point, waypoints to be followed and the destination point. Each waypoint is represented as in the input.

We discretize the space and consider it in 2D as group of triangular lattices while in 3D it is composed by tetrahedrons. A lattice is a whose drawing, embedded in some Euclidean space $\mathbb{R}^n$, that forms a regular tiling.

Figure 6 is an example of a triangular lattice.

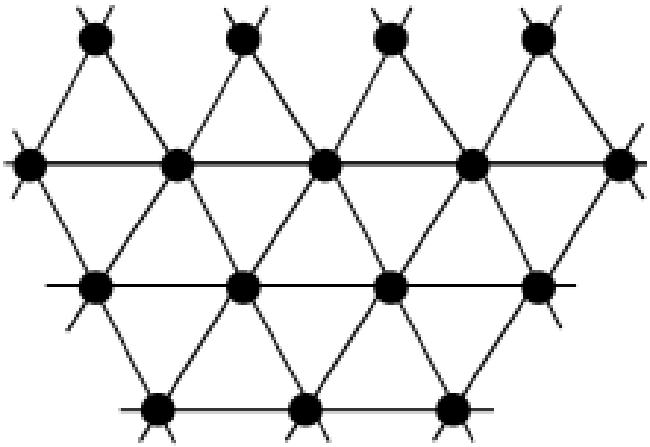

Fig. 6. Triangular lattice graph

We chose triangular lattices over quadrangular ones since in the former the distance between points approaches the Euclidean distance while in quadrangular ones the distance would be more similar to the Manhattan distance.

Using lattices, we can then extend the knapsack problem, by obtaining a network model and then obtaining the shortest path [7] where we set the objectives as the minimization of fuel consumption and travel time while the criterion space is set by the restrictions we receive as input.

When choosing the resolution of the lattice, we are posed with a problem. With lower resolutions, computation time will be lower since there are fewer arcs and nodes to compute the path to. However, the route can be sub-optimal because the distance from the obstacle will not be a true optimum since there will be a minimal safety distance in which the airplane distances itself from the obstacle plus the distance between nodes set by the resolution. On the other hand, with higher resolutions, the computation will be slower, however we will approach the true optimum since the safety distance will remain the same however the distance between nodes will be lower. In our solution, we will need to find a compromise between the resolution and the performance having in mind that the computation will need to be quick enough to be used online.

We reached the conclusion to use Captivo et al. labeling algorithm [7]. This algorithm is simple to use and becomes very efficient when using the optimizations proposed by [13] making it ideal for an online flight planner, ready to make updates if adversarial changes happen.

In our problem we will inevitably reach situations where we will have to choose a solution from this set. A potential solution is to use TOPSIS or AHP.

It is important to refer that our paths are not just in 2D meaning that a 2D graph is not enough to find the optimal path since altitude variations are possible, thus making the problem more difficult.

Overall, we intend to provide a solution that companies are able to pick up and adapt easily to their needs and their priorities, giving flexibility in the restriction set and their values. This is made possible by not restricting our algorithm to a fixed set of restrictions but instead make it dynamic and adaptive to sudden changes. For example, before taking off, the weather was clear, but after departure a weather-storm developed. Then it could be modeled as a restriction and from there we could make changes in the route. The other way around could also be used as an example, if a weather-storm was taking place and then it disappeared.

Our module will only act if conditions change, meaning that only if we receive a signal that a condition has changed, only then we would verify the current route and changed it if needed be.

For the development, we decided to split it into two phases. In the first phase we will disregard aircraft restrictions and only consider route restrictions to find a feasible trajectory. This makes it easier to test if a route is optimal from a given point since we only need to find a geometric path between two points having obstacles in the middle.

In the second phase we will include aircraft dynamics as restrictions, meaning that speed, rate-of-climbing, bank angles, all that limits aircraft movement will be used. This is more challenging, since we are dealing with objects moving at high speeds, meaning that variation in the route might be significant if the algorithm is not fast enough in its computation.

We plan to use the OODA cycle as guidance to what the solution is able to do. Basically speaking, the module observes that conditions were changed, makes an hypothesis on a new route (it could be the same route), tests it and then forwards it, sending feedback back to another modules on the change that happened.

This cycle could be used in pathfinding problem and specifically in our case because:

1. In the observing phase, the changing conditions would be the restrictions: whether they appear or disappear, they change the environment possibly making new routes.
2. If conditions change and new routes are possible, then it makes sense to verify effectively if a better route exists and if the current course (orientation) should be changed.
3. A different module would then perform a test on its feasibility and if it is feasible it would act upon it and change current course.

We will use an open-source dynamic model to test the feasibility of our generated routes in the given aircraft. This model has to be fast enough to validate the trajectories and discard the ones that are not feasible.

4 Evaluation Methodology

To evaluate our work we plan to use two benchmarks: optimality of the solution and time of computation.

The optimality of the solution is important to compare because as the distance to the optimum increases so does the overall cost meaning that if our solution is the most optimal then it gets difficult to argue against it. However, it is possible that our solution is optimal and not usable when an aircraft is already in flight because it is not efficient. And so we turn into our second benchmark, time.

Time is very important (if not the most) since our work intends to provide a efficient solution for airplanes to use in-flight and if we can not find one ideal solution efficiently then there are already very good flight-planners able to find optimal paths.

To test our solution we will first test its feasibility by running it through a simulator JSBSim and verify if everything is as it is supposed to. After this verification, we intend to compare the computation times of the routes generated and compare it to the times when generating routes with real flight-planners.

In terms of test data, JSBSim provides a way of generating routes with restrictions. What we can do is to provide an initial route to our module and set the restrictions manually and then compare the output with the one generated by JSBSim to verify if its optimal.

There is another flight simulator being produced by a colleague that has intention of being able to simulate flight routes via GPU acceleration, accepting routes as input. The objective of this work is to be able to simulate routes in real-time when airplanes are already airborne. Therefore we have intention of simulating our generated routes in this flight simulator as a way of testing feasibility.

5 Calendarization

Considering that the delivery is in the end of the month of October, we have about 9 months of time to do the thesis.

We divided the time into 4 phases: Solution research, solution development, results and experimentation and thesis writing.

Solution research is the initial phase where we gather all what we have learned with previous work, find implementations of said work and decide how we will implement the solution. This research should take about 1 to 2 months. However, we will always need to consult the work and investigate further, if there are solutions that we missed and/or different ways of tackling the problems we will be encountering along the way.

Solution development is the phase on where we develop our flight-planner and this phase is one that takes most of the time. We put it into 4-5 months of development time.

Results and experimentation is the phase on where we test and evaluate our flight-planner and compare it to other flight-planners. This phase should take 1-2 months.

The thesis writing phase is where we produce a document (the thesis) in which we describe what we have developed as solution, report the results and provide conclusions on what we have achieved and future work to be done in order to further improve our solution. This phase should take 1-2 months to be completed.

6 Conclusion

Today's flight planners are not able to provide real time updates to routes in case adversarial changes happen. Our work intends to provide a very fast and very reliable solution to a path-finding problem. We have to generate a new route in case conditions change and this route must be feasible, meaning that the airplane needs to able to complete this trajectory. While seemingly simple, this problem is very complex and has a lot of challenges to take into account. We need to make the program flexible enough, so that the restrictions can change, however we need to be able to provide a solution in a very short time, since we are dealing with entities that move at a very high speed and conditions vary very quickly, making so that in a time window of minutes, several routes could be and then stop being optimal.

References

1. Alonso-Portillo, I., M. Atkins, E.: Adaptive Trajectory Planning for Flight Management Systems. Tech. rep., University of Maryland, Maryland (2006). https://doi.org/10.2514/6.2002-1073
2. Babaei, A.R., Mortazavi, M.: Three-dimensional curvature-constrained trajectory planning based on in-flight waypoints. Journal of Aircraft **47**(4), 1391–1398 (2010). https://doi.org/10.2514/1.47711
3. Bazgan, C., Hugot, H., Vanderpooten, D.: Solving efficiently the 0-1 multi-objective knapsack problem. Comput. Oper. Res. **36**(1), 260–279 (Jan 2009). https://doi.org/10.1016/j.cor.2007.09.009, `http://dx.doi.org/10.1016/j.cor.2007.09.009`
4. Bellman, R.: On a routing problem. Quarterly of Applied Mathematics **16**(1), 87–90 (1958). https://doi.org/10.1090/qam/102435
5. Bousson, K., Machado, P.: 4D Flight trajectory optimization based on pseudospectral methods. World Academy of Science, Engineering and Technology **70**(May 2016), 551–557 (2010)
6. Canny, J., Reif, J.: New Lower Bound Techniques for Robot Motion Planning Problems. Annual Symposium on Foundations of Computer Science (Proceedings) pp. 49–60 (1987). https://doi.org/10.1109/sfcs.1987.42
7. Captivo, M.E., Clímaco, J., Figueira, J., Martins, E., Santos, J.L.: Solving bicriteria 0-1 knapsack problems using a labeling algorithm. Computers and Operations Research **30**(12), 1865–1886 (2003). https://doi.org/10.1016/S0305-0548(02)00112-0

8. Carpin, S., Pillonetto, G.: Merging the adaptive random walks planner with the randomized potential field planner. Proceedings of the Fifth International Workshop on Robot Motion and Control, RoMoCo'05 **2005**, 151–156 (2005). https://doi.org/10.1109/romoco.2005.201416
9. Carpin, S., Pillonetto, G.: Motion planning using adaptive random walks. IEEE Transactions on Robotics **21**(1), 129–136 (2005). https://doi.org/10.1109/TRO.2004.833790
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1), 269–271 (1959). https://doi.org/10.1007/BF01386390
11. Dougui, N., Delahaye, D., Puechmorel, S., Mongeau, M.: A light-propagation model for aircraft trajectory planning. Journal of Global Optimization **56**(3), 873–895 (2013). https://doi.org/10.1007/s10898-012-9896-1
12. Fallast, A., Messnarz, B.: Automated trajectory generation and airport selection for an emergency landing procedure of a CS23 aircraft. CEAS Aeronautical Journal **8**(3), 481–492 (2017). https://doi.org/10.1007/s13272-017-0252-5
13. Figueira, J.R., Paquete, L., Simões, M., Vanderpooten, D.: Algorithmic improvements on dynamic programming for the bi-objective {0,1} knapsack problem. Computational Optimization and Applications **56**(1), 97–111 (2013). https://doi.org/10.1007/s10589-013-9551-x
14. Ford, L., Fulkerson, D.: Maximal Flow Through a Network
15. Gomes Da Silva, C., Clímaco, J., Figueira, J.: A scatter search method for bi-criteria {0, 1}-knapsack problems. European Journal of Operational Research **169**(2), 373–391 (2006). https://doi.org/10.1016/j.ejor.2004.08.005
16. Hwang, C.L., Yoon, K.: Multiple Attribute Decision Making Methods and Applications, vol. 618 (1981)
17. Islami, A., Chaimatanan, S., Delahaye, D., Islami, A., Chaimatanan, S., Delahaye, D., Scale, L., Planning, T.: Large Scale 4D Trajectory Planning pp. 27–47 (2016)
18. Jouglet, A., Carlier, J.: Dominance rules in combinatorial optimization problems. European Journal of Operational Research **212**(3), 433–444 (2011). https://doi.org/10.1016/j.ejor.2010.11.008, `http://dx.doi.org/10.1016/j.ejor.2010.11.008`
19. Jun, M., D'Andrea, R.: Path Planning for Unmanned Aerial Vehicles in Uncertain and Adversarial Environments pp. 95–110 (2003). https://doi.org/10.1007/978-1-4757-3758-5_6
20. Orlin, J.B.: Ahuja, Magnanti, Orlin - Network flows Theory, algorithms and applictions (1993). https://doi.org/10.1016/0166-218X(94)90171-6
21. Paielli, R.A.: Trajectory Specification Language for Air Traffic Control. Journal of Advanced Transportation **2018** (2018). https://doi.org/10.1155/2018/7905140
22. Peter E. Hart, Nils J. Nilsson, B.R.: A Formal Basis for the Heuristic Determination of Minimum Path Costs (1968)
23. Saaty, T.L.: What is the analytic hierarchy process? (2011). https://doi.org/10.1007/978-1-4419-6281-$2_3$1
24. Sislak, D., Sislak, D., Volk, P., Volk, P., Pechoucek, M., Pechoucek, M.: Flight Trajectory Path Planning. 2009 ICAPS Scheduling and Planning Applications woRKshop (SPARK) pp. 76–83 (2009)
25. Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., Feyzeau, P.: Path planning: A 2013 survey. Proceedings of 2013 International Conference on Industrial Engineering and Systems Management, IEEE - IESM 2013 **2013**(October) (2013)
26. Zermelo, E.: ber die Navigation in der Luft als Problem der Variationsrechnung (1930)