Separating air traffic flows by allocating 3D-trajectories

David Gianazza, Nicolas Durand

SEPARATING AIR TRAFFIC FLOWS BY ALLOCATING 3D-TRAJECTORIES

David Gianazza and Nicolas Durand, CENA-LOG, Toulouse, France

Abstract

This paper introduces two algorithms which allocate optimal separated 3D-trajectories to the main traffic flows. The first approach is a *1 vs. n* strategy which applies an A* algorithm iteratively to each flow. The second is a global approach using a genetic algorithm, applied to a population of trajectory sets. The algorithms are first tried on a toy problem, and then applied to real traffic data, using operationnal aircraft performances. The cumulated costs of the trajectory deviations are used to compare the two algorithms.

Introduction

The airspace congestion over Europe is a critical problem, currently addressed by the Air Traffic Flow Management by allocating departure time slots to aircraft, thus generating costly ground delays. At a strategic level, traffic orientation schemes may be defined, with a set of routes assigned to each main origin-destination flow. The aim of these schemes is to balance the traffic among the network's routes and sectors. Modifications of the network may also be decided, in order to increase the overall capacity. In this context, airlines operators are free to choose a flight path and a requested flight level for each of their flights, on a day-to-day basis. A consequence is that many flights request the same flight levels and the most direct routes, thus increasing the airspace congestion. This system is now reaching its limits (c.f. Eurocontrol report [1] : a small amount of additionnal traffic generates a great increase in the cumulated delays) and the need for alternatives of higher capacity arises.

Numerous papers related to Air Traffic Management appear in the litterature, many of them dealing with capacity problems ([2], [3], [4]) or with dynamic flight planning through a congested airspace ([5], [6]), using a variety of deterministic or stochastic methods. Solving these problems is not in the scope of this paper, which deals with static 3D-trajectory design. The routes network design is adressed in [7], using Voronoï diagrams and clustering methods which iteratively move and merge the crossing points of the network. Although quite interesting, this approach is mainly bi-dimensionnal and does not take into account the vertical evolutions of aircraft. In [8] and [9], Graph Colouring techniques are used to assign cruising flight levels to aircraft flying on direct routes, in order to ensure vertical separation during the cruise. The climbing or descending trajectory segments are not considered. In [10], an interesting concept of TMA-to-TMA handover is assessed through statistical studies and traffic simulations. The idea is to remove a percentage of the traffic from the current ATC system (and from the slot allocation process) by defining conflict-free routes between the main terminal areas. Aircraft flying on these routes would have priority over the rest of the traffic and would be handled by specific departure and arrival management tools. Removing even a small percentage of traffic may drastically reduce delays, as long as the impact on the overall system capacity is limited. However, only horizontal separation between routes is discussed: crossing routes are either forbidden or allowed in a very limited way. So only a few traffic flows could be considered, without significant profit in terms of conflict reduction. In addition, it is not specified how aircraft would be sequenced on each route and how traffic flows separation would be achieved.

In this paper, we propose to reduce congestion by assigning static 3D-trajectories to the main traffic flows between Terminal Areas, using two alternative methods. The first one is an *iterative 1 vs. n* strategy where flows are considered in a chosen order (for example in decreasing order of size), and the second is a stochastic approach with a global strategy. With the first method, trajectories are sequentially computed, so that the first flows will have the most direct (closest to default) 3D trajectories. The second method searches a global optimum by applying a genetic algorithm to populations of trajectories sets.

The first part of the paper briefly describes a simplified model, the algorithms and their application to a toy problem. with aircraft following

linear climb or descent slopes. Only the main features, useful to the understanding of the rest of the paper, are presented. The reader may refer to [11] for the discussion of the problem complexity, the choice of the algorithms, and for more details on the A* and genetic algorithms.

In the second part, the two algorithms are applied to real traffic samples (France and Europe), with realistic aircraft performance models (the ones currently used in the operationnal system). Two new modes of trajectory interference detection, which were not presented in [11], are introduced. The results of the algorithms are compared by considering the cumulated cost of trajectory deviations. The conclusion summarizes the main results and gives an overview of the future work on the subject.

The toy problem

Trajectory model

Let us first consider a fairly simplified model for our trajectory design problem. The set of flows shall be arbitrarily chosen (origins and destinations on a grid, or on a circle, for example). A flow is defined as a set of flights between a departure airport and an arrival airport. The following simplifications are made : the airspace is considered as an Euclidean space, where all airports are at altitude 0. Latitudes and longitudes on the ellipsoïd earth surface are converted into (x, y) coordinates by a stereographic projection, and the altitude in feet shall be our z coordinate[1]. All aircraft fly with identical performances and follow linear slopes of climb and descent, and all aircraft belonging to a given flow i request a same cruise flight level , and follow by default a direct route between departure and arrival.

[1] This approximation is possible only as long as we stay in an area around the projection center which is not too large. It also introduces some errors in the computation of distances between trajectories : aircraft usually follow orthodromic routes over the earth surface, which will not be projected as straight lines on our stereographic plane. These errors can be balanced in our problem by increasing the separation minima between trajectories.

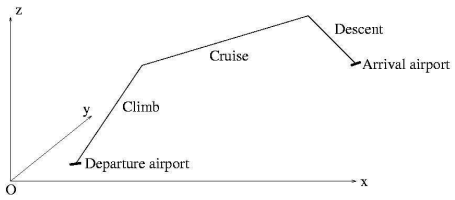


Figure 1: Basic model of a default trajectory

These simplifications allow us to allocate only one trajectory per flow. A trajectory shall be a sequence of line segments in our Euclidean space. Figure 1 shows a default trajectory between a departure airport and an arrival airport, with a climb towards a requested flight level RFL_i , and a descent downto the destination airport.

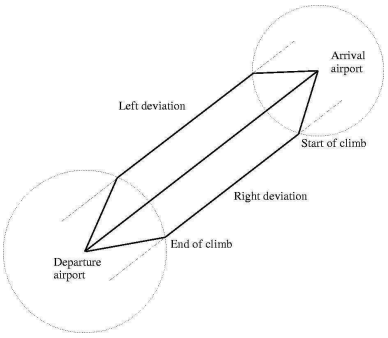


Figure 2: Horizontal deviations

In order to avoid conflicts between trajectories, we shall introduce some lateral or vertical deviations. Horizontally, only three possibilities are allowed, as shown on the figure 2 : the direct route, or left or right parallel routes. The radii around the departure and arrival airports, and the value of the offset, are input parameters. Vertically, we shall allow a succession of different flight levels (CFL stands for Cleared Flight Level), as illustrated on figure 3. Each of the flight levels shall be between a minimum flight level FL_{min} and the requested flight level RFL. The vertical deviations are then characterized by a sequence of pairs (d_j, CFL_j), where d_j is the distance (along the route) at which the vertical evolution towards CFL_j begins.

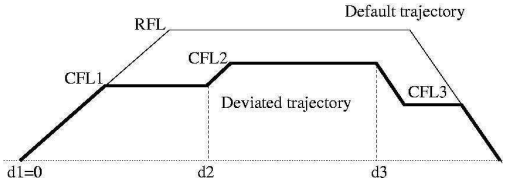


Figure 3: Example of a vertical profile.

Each trajectory is then completely represented by the following variables :

- the choice of a route, represented by a variable r (equal to 0 for the direct route, 1 when the deviation is to the right, and -1 when it is to the left),
- and the sequence of pairs (d_j, CFL_j).

Cost of a trajectory

We are now able to define a cost related to the lateral and vertical deviations. The cost of a vertical deviation depends on the surface between the effective vertical profile and the cruise level :

$$L_i \times RFL_i - surface(profile_i)$$

where L_i is the length of the chosen route. This cost should not depend on the distance between origin and destination (otherwise small deviations on long flights may cost as much as big deviations on short flights), so we shall divide this expression by the route length L_i. The cost of a lateral deviation depends on the route elongation $(L_i - Lref_i)/Lref_i$, where $Lref_i$ is the length of the direct route.

Finally, the total cost of a trajectory $cost(i)$ is a combination of the two, with K a chosen factor:

$$RFLi - surface(profile_i)/L_i + K \times (L_i - Lref_i)/L_i$$

Detecting interfering trajectories

Let us now define the notion of *interference* between trajectories. We will deliberately avoid to use the term *conflic*t, which in the aviation community refers to the fact that two aircraft are (or will be) closer than the allowed separation minima (usually 5 nautical miles horizontally and 1000 feet vertically). As trajectories are sequences of 3D line segments, we shall not use the horizontal or vertical distance between points in space.

In [11], an normalized distance $d(a, b)$ equal to the square root of $(x^2+y^2)/N_h^2+z^2/N_v^2$ was introduced (with (x, y, z) the coordinates of the vector between points a and b, and N_h and N_v the standard horizontal and vertical separations). Two trajectory segments $s_1 \in T_1$ and $s_2 \in T_2$ are said to be interfering when the closest points (using the distance d) are at a distance less than $\sqrt{2}$, when at least one segment is climbing or descending. When both segments are at a constant altitude, they will interfere when the altitude difference is less than N_v and when their horizontal separation is closer than N_h. The use of this new distance d brings an additional margin in the separation of trajectories when compared the exact definition of a conflict, but the implementation of the detection of interferences is much easier this way.

Problem description

Let us consider a set of N traffic flows. For each flow there is a default trajectory between departure and arrival. These trajectories may interfere one with each other, in the sense of the above definition. Our goal is to find a set of N new trajectories, as close as possible to the default trajectories, but separated in space according to the distance criterion defined in the previous subsection. This is a minimisation problem, for which we may choose among two strategies.

A first approach, that we will call *1 vs. n*, is to consider each flow in turn and try to find a new trajectory of minimal cost, separated from the previous n trajectories ($n < N$). With this strategy, the solution found (when there is one) will depend on the order in which the flows are considered. Different sequences will lead to different solutions, and the flows at the beginning of the sequence shall obtain more direct trajectories than the ones at the end. As we are trying to find the shortest path through the airspace while avoiding obstacles (the other trajectories), using an A* algorithm seems adequate.

The second strategy, that we may call *global strategy*, consists in minimizing a cost associated to the set of trajectories, without any constraint of order on the flows. The global cost should cumulate the costs of each individual trajectory. So there are in fact two different problems, depending on the chosen solving stragegy : *global* or *1 vs. n*.

The complexity of the global problem has already been discussed in [11]. The search space is split in a huge number of connected components (more than thirty thousand billions for 10 trajectories), because of the many possible combinations that can be used to separate trajectories. This huge number and the fact that we

don't know *a priori* in which component(s) is (or are) the optimal solution(s), make the global problem highly combinatorial and prevent to use local methods. Furthermore, when the number of trajectories increases, we have to face big size problems which are generally difficult to handle with deterministic optimization (as A* or Branch & bound for example), so we will use a genetic algorithm for the global problem.

The A algorithm*

The A* algorithm is applied iteratively to each flow. Its aim is to find the shortest trajectory from departure to arrival, while avoiding the already computed trajectories.

The idea of the A* algorithm (cf. [12]) is to search the best path through a tree of possibilities, restarting at each step from the best possible node encountered so far during the search. To do this, we need a cost function for the transitions between states (tree nodes), and a heuristic function which shall estimate the cumulated cost of the transitions remaining between the current state and a possible solution. In our problem, the states shall represent choices in the possible deviations (horizontal or vertical), made at each step of the trajectory. The cost and heuristic functions depend on the extent of the trajectory deviations, and are detailed in [11].

As discussed before, a trajectory is completely defined by the choice of a route *r*, and a list of constraints on the vertical profile (d_j, CFL_j). The choice of a distance d_j is not free: it can only be a multiple of one tenth of the route length. There is also an additional implicit constraint : the vertical profile must end at the arrival airport, following the final descent slope. A trajectory state is simply represented by $(r, \{(d_j, CFL_j) / j \in [0, 9]\})$. As an example, the default trajectory will be *(0,[(0,RFL)]*). The trajectory is built step by step : climb (or descend) towards CFL_j then stay levelled until you climb or descend towards CFL_{j+1}. So a node is completely defined by the trajectory, the index *j* and the current vertical evolution *evol*, which may be either *Climb* or *Descent* (during the evolution towards the current CFL_j), or *Levelle*d.

In order to search among the tree of all possible nodes, we need to define rules that generate new nodes : the sons of the current father node. In our case, the sons are the alternatives for the next trajectory step. If an interference is detected between the next step and previous trajectories, the corresponding son is discarded. This tree-search mechanism is detailed in [11]. The search ends when the arrival airport is reached. The trajectory built by the A* is then the one closest to the default trajectory that does not interfere with the other trajectories.

The genetic algorithm

Genetic algorithms are based on the paradigm of natural evolution. Optima are reached through a process of crossing, mutation and selection of the fittest individuals. This process is applied to a population of chromosomes. The reader may refer to [13] and [14] for an overview of the latest algorithms based on the evolutionnary paradigm, or to [15] and [16], [17] for more details on genetic algorithms. A good state of the art of optimization using genetic algorithm may also be found in [18] and [19], with a practical application to the Air Traffic Control domain (specifically to conflict solving) in the latter. In our problem, a chromosome will be a set of n trajectories. Each chromosome of the initial population is generated by randomly choosing its n trajectories (within given bounds concerning the flight levels).

The *fitness* criterion allowing to select the best trajectory sets at each step is directly related to the cumulated cost of trajectory deviations. However, the fitness function also takes into account the interferences between trajectories : chromosomes with interfering trajectories shall be penalized. Let us first define a triangular matrix *C*, which diagonal values indexed *(i, i)* are the deviation costs of each trajectory. The other values, indexed *(i, j)* with $i \neq j$ store the number of interfering segments between trajectories T_i and T_j.

$$\begin{array}{rll} & C_{ii} = & cost(i) \\ \forall i \neq j & C_{ij} = & \sum_{(s_p, s_q) \in \mathcal{T}_i \times \mathcal{T}_j} \delta(s_p, s_q) \end{array}$$

$$\delta(s_p, s_q) = \begin{array}{l} 1 \text{ if } d(s_p, s_q) \leq \sqrt{2} \\ 0 \text{ otherwise.} \end{array}$$

Let us note *f(i)* the sum of the matrix elements *C(i,j)*, with $i<j$. This sum is equal to zero when

trajectory T_i is not interfering with any other trajectory. For a chromosome with completely separated trajectories, the sum of *f(i)* for $i \in [0,n]$ will be equal to zero. The fitness criterion F of a given chromosome (a set of n trajectories) is defined by :

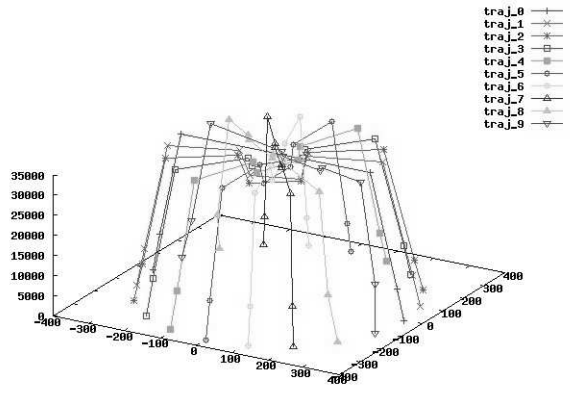
$$\mathcal{F} = \begin{cases} 1 + \frac{n}{1+\sum_i C_{ii}} & \text{if } \sum_i f(i) = 0 \\ \frac{1}{\sum_i f(i)} & \text{if } \sum_i f(i) > 0 \end{cases}$$

Fitness values shall then be less than 1 for chromosomes with interfering trajectories, and above 1 for chromosomes with separated trajectories. In this last case, the smaller the trajectory deviations will be, the greater the fitness will be.

The crossover operator and the mutation operator (which re-uses the A* algorithm) are detailed in [11].

Results for the toy problem

The detailed results of the application of the algorithms to the toy problem may be found in [11]. For a semi-circle configuration, with flows converging towards a single point and requesting a same flight level (FL350), the A* algorithm computes non-interfering trajectories with a cumulated deviations cost of 52.068. The computation time on a Xeon CPU 2.8 GHz is 5.76 seconds.



Figures 4 and 5 show the best solution found by the genetic algorithm. We see that the lateral deviations of trajectories 8 and 9 allow slighly different vertical deviations for these trajectories than in the A* solution, thus allowing to separate the trajectories using 8 flight levels instead of 10. This solution, with a *fitness* of 1.1901 and a cumulated cost of 51.62 is better than the one found by the A*. The computation time (on a Xeon CPU 2.8 GHz) is 1033.13 seconds.

Figure 4: GA best solution for 10 flows on a semi-circle

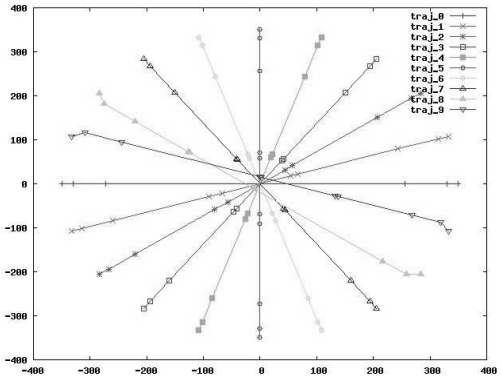


Figure 5: 2D view of the GA best solution for 10 flows on a semi-circle

Application to real traffic data

The application of the algorithms to real traffic data implies several modifications in the model, in the detection of interfering trajectories, and in the algorithms themselves. But let us first have a closer look at the traffic flows over France and Europe, in order to get a better view of the problem to solve.

Traffic flows over France or Europe

Traffic flows are here defined as flows of aircraft flying between two points. These points may be departure or arrival airports, or entry or exit points in the airspace. Note that this definition is more restrictive than the informal definitions usually found in the Air Traffic Management community, where traffic may flow from one geographic area to another, or through a given sector, or over a chosen point or route segment.

Figure 6 shows the origin-destination flows of 10 flights or more in France, on the 21st June of 2002. Figure 7 shows the flows of 10 flights or more in Europe, on the same day. Let us note that our definition of Europe is here quite large : it is in

fact the area described in the Eurocontrol airspace database.

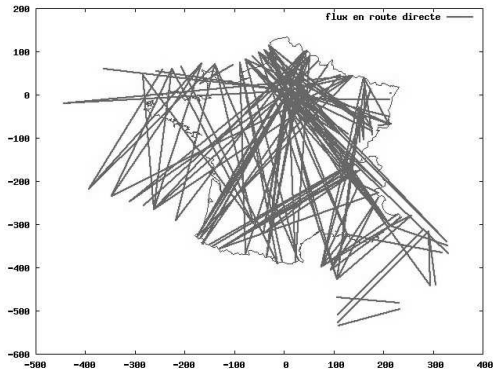


Figure 6: Flows of 10 flights or more, in the french airspace (21st june 2002)

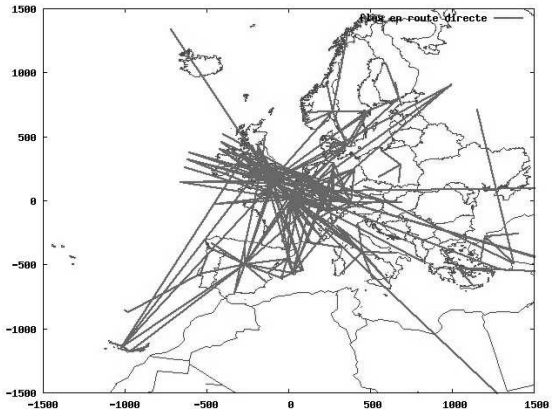


Figure 7: Flows of 10 flights or more, in the european airspace (21st june 2002)

A short traffic analysis was made on a single day of traffic (21st june 2002). The reader may refer to the TOSCA study ([10]) for more exhaustive statistics, over several months. There were 29687 flights over Europe on June the 21st of 2002, flying on 11313 different origin-destination flows. By sorting the flows by decreasing order of size, we are able to compute the cumulated traffic for flows of n flights or more. The 1567 flows of 5 flights or more represent 46 percent of the total traffic. The 392 flows of 10 flights or more represent 20.8 percent of the traffic, and when considering the 74 flows of 20 flights or more, we are only at 6.9 percent of the total traffic. As a consequence, we will have to consider a great number of flows if we want to find trajectories for a significant amount of traffic. However, there is no interest in considering too many flows for our problem : more than a half (5865 to be precise) of the 11313 flows comprise only one flight. It makes no sense to assign permanent, static 3D-trajectories to such flows.

These considerations give support the alternative proposed in the TOSCA study (where similar results can be found), which consists in grouping the airports in large terminal areas. The traffic flows are then defined between these big TMAs. This allows to handle a larger amount of traffic with a smaller number of flows, but it implies to organize the traffic within each TMA, or between adjacent TMAs. For the moment, we will only adapt the proposed algorithms to flows between airports (or entry/exit waypoints), considering that removing only a relatively small amount of trafic from the slot allocation process may lead to high profits in terms of delay reduction, as shown in the TOSCA study.

Another interesting figure, concerning european data, is that most flights (more than 85 percent on the considered day) are intra-european : they take-off and land within the considered airspace.

Table 1: Traffic per type of origin-destination (D= Departure airport ; A= Arrival airport ; W= entry or exit Waypoint)

	Total	D->A	D-> W	W -> A	W -> W
Flows	1826	832	328	323	343
p.c. of flows	100 %	45,6 %	18,0 %	17,7 %	18,8 %
Flights	8540	2028	1563	1601	3348
p.c. of flights	100 %	23,7 %	18,3 %	18,7 %	39,2 %

Let us now focus on the french airspace, which is quite a different problem. There were 8540 flights on the 21 June 2002, flying on 1826 origin-destination flows. Among these flights, 39.2 percent are overflying traffic, representing only 18.8 percent of the flows, as shown on table 1, and only 23.7 percent are purely domestic, although they represent nearly half of the flows (45.6 percent). International flights (columns D->W and W->A) represent 37 percent of the traffic. So we see that more than a half of the origin-destination flows begin and/or end at the border of the french airspace, and more than 75 percent of the traffic comes from or goes to a foreign airport.

Further analysis shows that the 77 flows of 20 flights or more already represent 42.4 percent of the

total traffic. The 166 flows of 10 flights or more represent 56 percent, and the 371 flows of 5 flights or more represent 72.1 percent When comparing to the european data, we see that we need to consider a relatively small number of flows in order to handle a significant part of the traffic. The reason for this is that the usual entry and exit points, issued from the actual routes network, were used in the flows computation. If we had re-computed the entry and exit points for each flight, following a direct route between departure and destination (which could be outside the French airspace), we would most probably have a much greater number of entry and exit points. The traffic would consequently be dispatched on a greater number of flows.

Problem discussion

The brief study described in the previous section shows that considering the whole european airspace or only a part of it (namely the french airspace) leads to completely different flows configurations. In the first case, we will have to deal mostly with intra-european flows, and the problem will be quite close to the toy problem of the first part of this paper. In the other case, the proportion of traffic entering or exiting via a waypoint at the border is much higher, and we will have to take this into account in the algorithms: they should be able to allocate separate entry or exit flight levels to flows beginning or ending over a same waypoint.

Trajectory model

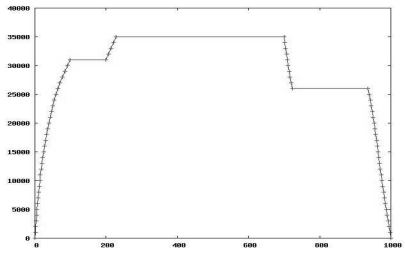


Figure 8: Typical vertical profile (A320)

The trajectory model for the toy problem was rather simple, with linear climb and descent slopes. To be more realistic, it should take the real aircraft performances into account, and also the fact that there may be several types of aircraft flying within a given origin-destination flow, with possibly a great disparity of performances.

Figure 8 shows an example of nominal flight profile for an airbus A320, computed with the performance model currently used in the european ATFM systems. A consequence of the disparity of climb or descent profiles for different aircraft is that a sequence of constraints *at distance* d_j, *start a vertical evolution towards* CFL_j, on a given origin-destination flow, will generate several flight profiles. This is illustrated on figure 9 which shows such profiles (for the following aircraft types: A340, B742, B743, B744, B762, B763, B772, DC10, L101, MD11).

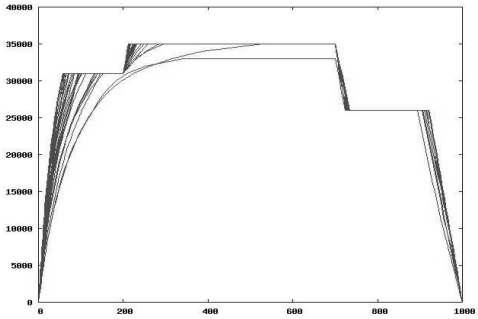


Figure 9: Example of several flight profiles on a same flow

In order to be as realistic as possible, a vertical uncertainty zone is defined for each 3D trajectory: it is the hull of all the possible flight profiles for all types of aircraft of the flow. In order to avoid the computation of all these profiles at each step of the algorithms, the upper and lower hulls of the climb and descent profiles are pre-computed for each flow, and then used by the algorithms to compute the uncertainty zone when necessary.

Detection modes

For the toy problem, the detection of interfering trajectories used a distance d which added a margin to the separation standards. The false detection of interferences is the drawback of this method. These false detections may lead to unnecessary trajectory deviations, as illustrated on figure 10. In addition, this distance criterion is unable to cope with the vertical uncertainty zones described above.

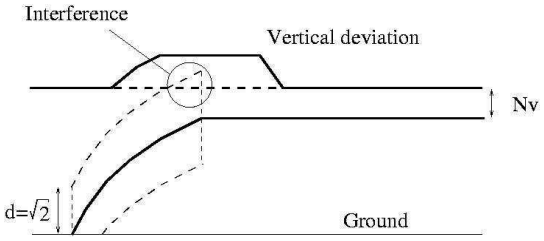


Figure 10: Example of false detection and unnecessary deviation, using distance *d*

To avoid these problems, a new type of detection is introduced. A tube is defined around each trajectory segment, using the separation standards and the vertical uncertainties, when needed. When there is no uncertainty the tube has a square section, of width N_h and height N_v , the 3D segment being at the exact center of it. When a vertical uncertainty is used, we simply add a margin of N_v^2 to the upper and lower bounds of the uncertainty zone. Two trajectories T_1 and T_2 are interfering if and only if there exists two segments $s_1 \in T_1$ and $s_2 \in T_2$ for which the corresponding tubes intersect. The algorithm detecting if two 3D tubes intersect is less easy to implement than the one with the distance *d*, but it provides a more accurate detection.

There is another problem which must be addressed for both types of detection. It is the case of two flows ending or beginning at the same airport. The interference between two such flows cannot be avoided (aircraft have to take-off or land at the same airport) but they are supposed to be handled by the approach control or the tower control. So the detection is turned off in the close vicinity of the airport: the trajectory segments located within a given range (about 15 NM for example) around the airport are not taken into account.

However, these no-detection zones do not solve all problems. For example, it may be useful to define several 3D trajectories for a given origin-destination flow, with different preferential cruise levels. An aircraft may then choose the most adequate trajectories, depending on its performances. In this case, it would be useful to consider that the two trajectories assigned to the same flow do not interfere during the initial climb and the final descent. The intermediate flight segments should of course remain separated.

Another problem, close to this one, is the case of outbound flows starting at an airport and going into very close directions. There is few hopes to solve the trajectory interferences with lateral deviations when the number of flows exceeds three (there are only 3 possible offset routes).

A solution is to increase the radius of the no-detection zone. But we would have to choose fairly extended zones, and the interferences between inbound flows, descending towards an airport, and outbound flows would not be detected. The other solution, which have been retained, is to consider that the initial climbs from a same airport never interfere one with each other, and the same for the final descents towards a same airport. The underlying idea is that the traffic separation should be ensured by a time sequencing, and not by statically separating 3D trajectories. This sequencing should be made by the departure or arrival managers. Let us note that with this solution, we will still continue to detect and solve trajectory interferences between departures and arrivals, outside the no-detection zone.

Let us summarize what have been written on the detection of interfering trajectories. There are three possible modes of detection :

- DIST: between 3D-line segments, with the distance *d*,
- ITUBES: between 3D-line segments, with the intersection of tubes of height N_v , and of width N_h ,
- IZONES: between uncertainty zones, with the intersection of tubes defined around these zones.

In addition, a no-detection zone is defined around each airport. The radius of the zone is an input parameter of the program (15 NM as default), The interference detection is turned off when considering two initial climbs from a same airport, or two final descent downto a same airport.

Let us note that the two first detection modes cannot be used when they are several types of aircraft in a given flow. We have to consider that there is only one type of aircraft per flow. The third mode is the most realistic.

Adapting the algorithms to the real problem

Concerning the A* algorithm, the transition cost and the heuristic are not modified. Only the rules of the nodes computation change, when there is an entry over a waypoint instead of a take-off, or an exit instead of a landing. For a father node corresponding to an entry at a given flight level, an additional son is computed, corresponding to an alternative entry at a the next lower flight level for which there is no interference at the entry, if it exists and if it is above a minimum flight level. The entry should be followed by a levelled flight segment of minimum length L_e . In the case of an exit, a similar rule is applied, with also a minimum length L_x for the levelled segment before exit.

The genetic algorithm is modified as follows. The entry and exit flight levels are randomly chosen, if necessary, when generating the initial population of trajectory sets. The mutation operator uses the modified A* algorithm, to find out an interference-free trajectory in replacement of a randomly chosen trajectory. It also may introduce random noises in these entry/exit levels as well as in the other cruise levels. The crossover operator also takes into account the new entry/exit levels. The fitness criterion remains unchanged.

Results for France

The A* is applied iteratively to the flows in the following order : the flows with the higher levels are considered first (reverse lexicographic order on the entry, exit, and cruise levels). For flows having the same levels, the most important flow is considered first.

Table 2 shows the results of the A* algorithm for flows of 10 flights or more, requesting a flight level above FL145, over France on the 21st June 2002. Note that the 71 flows represent 39.6 percent of the whole traffic. The value of the fitness criterion F is given for each detection mode. Although is is normally used only in the genetic algorithm, it was also computed for the A* solutions, because it is a good indicator of the number of interferences and of the trajectory deviations cost. Let us remind that it is related to the number of interfering segments when there are interferences (values smaller than 1), and to the deviation costs otherwise (values greater than 1).

Table 2: A* results for France

Detection mode	**DIST-A320**	**ITUBES-A320**	**IZONES**
Fitness F	1.238564	1.272030	0.083336
Nb. Fail	0	0	1
Time (s.)	134.17	73.51	106.95
Cost	296.64071	260.02877	(205.21511)
Nb. FLs above 145	19	19	20
Nb. FLs below 145	0	0	1
Route elong.	0.67 %	0.60 %	0.14 %

We see that for the IZONES detection mode, the algorithm found no solution for one of the trajectories. This is not due to the chosen mode : for other samples on other days, the A* may find a solution for each mode, or fail with another mode. The reason is simply that there is no solution when optimizing each trajectory in turn, with the chosen sequence of flows.

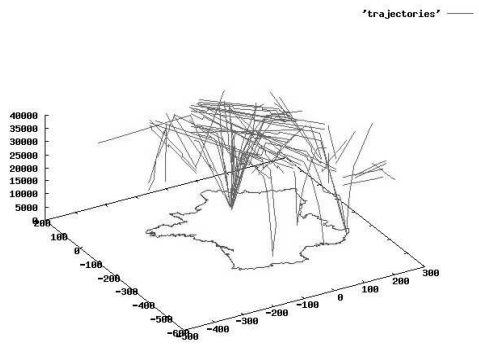


Figure 11: 3D view of the A* solution, for flows of 20 flights or more over France

The detection using the intersection of tubes (ITUBES-A320) seems more efficient than the one using the distance d : the fitness is better, and the cumulated cost of the deviations is 260.03, which is smaller that the 296.64 found for the DIST-A320 mode. The trajectories found by the A* algorithm (with DIST-A320 detection mode) are shown on figure 11.

The genetic algorithm was first run using the mutation operator of the toy problem (which calls the A* algorithm), with two different seeds for the random generator, and with the DIST-A320 mode

only . With a population size of 400 elements, evolving during 250 generations, with a crossover probability of 0.6, and a mutation probability of 0.1, the results are better than for the A* algorithm (1.283677, and 1.279077), but the computation times are quite longer : 32 hours 38 minutes 52 seconds for the first run, 22 hours 55 minutes 15 seconds for the second one.

When investigating the reason for this, it was found that the use of the A* algorithm in the mutation operator was the cause of these long calculation times. In the context of this operator, the A* is applied to a randomly chosen trajectory, and tries to find an optimal path through the whole rest of the traffic. The trajectories are initially generated at random. As a consequence, it is more difficult to find a way through this whole traffic than when applying the A* sequentially to a list of flows, beginning with the higher levels. An alternative mutation operator was tried, using a local search instead of the A* algorithm. It exhibited poor results in finding out a solution.

Table 3: GA results for France

Detection mode	**DIST-A320**	**ITUBES-A320**	**IZONES**
Fitness *F*	1.282858	1.315365	1.300095
Nb. Fail	0	0	0
Time (s.)	23761.60	24831.51	66970.20
Cost	250.03534	224.15867	235.61843
Nb.FL above 145	18	20	19
Nb. FL below 145	0	0	1
Route elong.	0.80 %	0.40 %	0.53 %

Finally, the alternative which seems to give the best results is an operator which randomly chooses (with given probabilities) among three mutation strategies : introduce random noises into the trajectories, run a local search algorithm which tries to improve the solution when there remains no interference, or use a modified A*. The A* is modified in order to use as input only the trajectories interfering with the initial trajectory, for the mutations occuring in the first generations. When the generation number is above one fourth of the maximum number of generations, the full A* is used.

Results for Europe

Table 4 shows the results of the A* algorithm for flows of 10 flights or more, requesting a flight level above FL145, over Europe on June the 21st of 2002. The 65 flows represent only a fairly small part of the overall traffic (6.20 percent). Table 5 shows the result of the genetic algorithm for the same problem, with a population of 400 elements, evolving over 300 generations, with the same crossover and mutation probabilities as before.

Table 4: A* results for Europe

Detection mode	**DIST-A320**	**ITUBES-A320**	**IZONES**
Fitness F	0.010204	1.112137	1.120674
Nb. fail.	2	0	0
Time (s.)	139.35	53.63	506.06
Cost	(566.08163)	587.59932	545.95383
Nb. FLs above 145	19	17	23
Nb. FLs below 145	0	1	0
Route elong.	1.89 %	1.90 %	0.68 %

The A* find no solution for one flow, when using the DIST-A320 detection mode. The genetic algorithm finds a solution in every case, but with longer computation times. When analyzing the log file of the GA runs, we observe that the algorithm found a solution in only 10 generations for the European problem (with DIST-A320 mode), whereas it needed 85 generations for the French problem. It seems easier to find solutions for the flow configuration over Europe, with most traffic taking-off and landing within the considered airspace, than for the one over France where most flows are international.

Figures 12 and 13 show the solution found with the IZONES detection mode. We see that the configuration of the flows is completely different from the flows in France: it is more star-shaped, with offset routes for flows in opposite directions between two airports. This leads to the conclusion that there may be some profit in splitting the whole problem into several independant sub-problems before running the genetic algorithm.

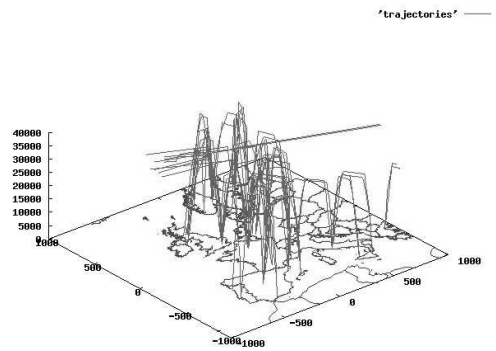


Figure 12: 3D view of the A* solution, for flows of 20 flights or more, over Europe

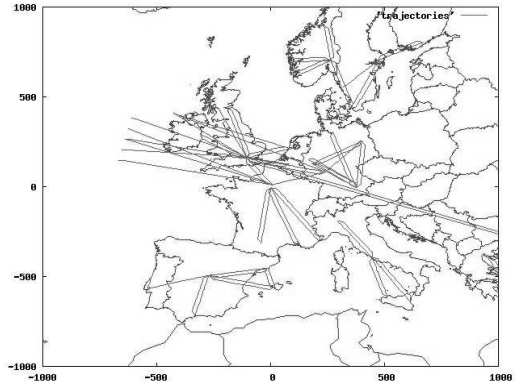


Figure 13: A* solution, for flows of 20 flights or more, over Europe

Table 5: GA results for Europe

Detection mode	**DIST-A320**	**ITUBES-A320**	**IZONES**
Fitness F	1.112151	1.112542	1.119510
Nb. Fail.	0	0	0
Time (s.)	50309.76	43440.69	204629.79
Cost	587.52588	585.48004	551.29116
Nb. FLs above 145	22	20	21
Nb. FLs below 145	0	0	0
Route elong.	2.04 %	1.97 %	1.22 %

Conclusion

As a conclusion, we may say that both algorithms exhibit good results on the toy problem as well as on more realistic problems, with traffic flows over France or Europe. The *global strategy* generally finds better results with the genetic algorithm than the 1 *vs.* n strategy with the A*. This is not a surprise, as the latter does not aim at solving the global problem.

There still remains to validate the results, by comparing the number and nature of conflicts between a traffic of reference and a traffic flying on the static 3D-trajectories found by the algorithms. This validation, using CATS/OPAS traffic simulator, is now under course and already shows good results on the French problem. It will be the subject of a future paper.

Further work may deal with the improvement of the genetic algorithm, by splitting each problem (a set of flows) into several sub-problems (flows interfering one with each other). Other improvements may be the definition of large TMAs, in order to handdle more traffic with less flows, in the European context. Another point to take into account, if we want to be more realistic, is the variety of aircraft flying on each origin-destination flow, with different flight performances. It could be useful to allocate several 3D-trajectories per flow . Each aircraft may then choose the most fitted trajectory, according to its flight performances. A last developpement, which is already under course, consists in introducing the time dimension in the algorithms, in order to make some pre-tactical 4D-trajectory planning for trains of aircraft having similar performances.

Biography

David Gianazza graduated from the (ENAC) in 1989 as an After beginning his career in the operational field in Brest ATC center, he obtained a master degree in computer science in 1996, and became deputy head of the division *Computer-assisted control tools* of the technical service STNA. He is currently preparing a PhD thesis at the Global Optimization Laboratory (LOG), a laboratory supported by CENA and ENAC.

Nicolas Durand graduated from the Ecole Polytechnique de Paris in 1990 and from the Ecole Nationale de l Aviation Civile (ENAC) in 1992. He has been a design engineer at the Centre d Etudes de la Navigation Aérienne (CENA) since 1992 and holds a Ph.D. in computer Science (1996).

References

[1] Performance Review Commision. Performance review report, an assessment of air traffic management in europe during the calendar year 2000. Technical report, Eurocontrol, 2001.

[2] D. Bertsimas and S. Stock Patterson. The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach. *Transporation Scienc*e, 34(3):239–255, August 2000.

[3] D. Delahaye and A. Odoni. Airspace congestion smoothing by stochastic optimization. In *Evolutionary Programming V*I, 1997.

[4] N. Barnier, P. Brisset, and T. Rivière. Slot allocation with constraint programming: Models and results. In *Proceedings of the fourth USA/Europe Air Traffic Management R&D Semina*r, 2001.

[5] M.R. Jardin. Real-time conflict-free trajectory optimization. In *Proceedings of the fifth USA/Europe Air Traffic Management R&D Semina*r, June 2003.

[6] C.H.M. van Kemenade, C.F.W. Hendriks, H.H. Hesselink, and J.N. Kok. Evolutionnary computation in air traffic control planning. In *Proceedings of the Sixth International Conference on Genetic Algorith*m. ICGA, 1995.

[7] Karim Mehadhebi. A methodology for the design of a route network. In *Proceedings of the Third Air Traffic Management R & D Seminar ATM-2000*, Napoli, Italy, June 2000. Eurocontrol & FAA.

[8] Vincent Letrouit. *Optimisation du réseau des routes aériennes en Europ*e. PhD thesis, Institut National Polytechnique de Grenoble, 1998.

[9] Nicolas Barnier and Pascal Brisset. Graph coloring for air traffic flow management. In *CPAIOR'02: Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problem*s, pages 133–147, Le Croisic, France, March 2002.

[10] L. Maugis, J.-B. Gotteland, R. Zanni, and P. Kerlirzin. TOSCA-II - WP3: Assessment of the TMA to TMA hand-over concept. Technical Report TOSCA/SOF/WPR/3/03, SOFREAVIA, 1998.

[11] D. Gianazza, N. Durand, and N. Archambault. Allocating 3d-trajectories to air traffic flows, using A* and genetic algorithms. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automatio*n, 2004.

[12] Judea Pearl. *Heuristic*s. Addison-Wesley, 1984. ISBN: 0-201-05594-5.

[13] T. Baeck, D.B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1 : Basic Algorithms and Operators.* Institute of Physics Publishing, 2000.

[14] T. Baeck, D.B. Fogel, and Z. Michalewicz. *Evolutionary Computation 2 : Advanced Algorithms and Operators.* Institute of Physics Publishing, 2000.

[15] J.H Holland. *Adaptation in Natural and Artificial System*s. University of Michigan press, 1975.

[16] David Goldberg. *Genetic Algorithm*s. Addison Wesley, 1989. ISBN: 0-201-15767-5.

[17] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learnin*g. Addison Wesley, 1989.

[18] Yann Le Fablec. Optimisation par algorithmes génétiques parallèles et multi-objectifs. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1992.

[19] Nicolas Durand. *Optimisation de trajectoires pour la résolution de conflits aériens en rout*e. PhD thesis, Institut National Polytechnique de Toulouse, 1996.

Email Addresses

gianazza@recherche.enac.fr

durand@recherche.enac.fr