Direct Route Optimization for Air Traffic Management Improvement

Andrea Pasini, Philippe Notry, Daniel Delahaye

Direct Route Optimization for Air Traffic Management Improvement

Andrea Pasini*, Philippe Notry†,Daniel Delahaye†
*ISAE-SUPAERO, Université de Toulouse, 31055 Toulouse, FRANCE
Email: andrea.pasini@student.isae-supaero.fr
†ENAC, Université de Toulouse, 31055 Toulouse, FRANCE
Email: philippe.notry@enac.fr
‡ENAC, Université de Toulouse, 31055 Toulouse, FRANCE
Email: delahaye@recherche.enac.fr

***Abstract*—Aircraft are able to connect any pair of points on earth by using direct routing thanks to the flight management system (FMS) which is the on-board system in charge of the navigation. From an air traffic control point of view, such direct routing is only possible in low density areas. When demand is high, aircraft have to stay on a route network for which crossing points are well located and do not change with time. This location stability, helps controllers to manage efficiently the traffic and ensure a higher capacity compared with management of the same traffic with direct routes for which crossing points always change in space and in time dimensions. This instability reduces the overall capacity of sectors. During the night, when traffic demand decreases, air traffic controllers often deliver direct routes to aircraft because they know that such aircraft will not interfere with other traffic. During the day, such direct route assignments are not proposed anymore and the controller keeps the aircraft on the route network in order to structure the traffic and to reduce the associated complexity. In this research we propose an intermediate milestone for which direct routes could be proposed to aircraft during the day when it is possible and this in order to reduce also the congestion at some points in the airspace. We then consider a real crossing in the French airspace where aircraft have to merge at the same point and exit toward different directions. An optimization algorithm has been developed for this crossing in order to minimize the workload of the controller in charge of this airspace. Three decision variables are then assigned to each aircraft: speed regulation, flight level setting and direct route setting. The associated objectives of this problem are the minimization of the conflict between aircraft (has to reach zero), the maximization of the number of aircraft having a direct route (avoiding the crossing) and the minimization of the flight deviation (speed and FL). When a direct is given to an aircraft, we must ensure that such aircraft is separated from other aircraft by one when they share the same spatial location; this ensure that the controller in charge of such aircraft has few monitoring workload for this aircraft (because it has no interaction with other traffic). This optimization must satisfy some constraints: speed and FL have to stay in a given range. This algorithm has been applied successfully to a test case quite similar to one located in the west of France ACC (Brest) for which the capacity of the crossing point has been strongly increased thanks to many direct routes that have been given.**

Keywords: Air Traffic Management, Air Traffic Optimization, Direct Routes, Simulating Annealing.

I. Problem description and mathematical model

A. Problem description

This study is focused on the decongestion and conflict resolution of air traffic over a waypoint in order to improve the Air Traffic Management. An ad hoc algorithm has been implemented and tested with a fictitious example, as presented in the following. The first application to solve real traffic cases will be done at the RESMI waypoint that is often overcrowded around flight level FL370. At this altitude the airspace is controlled by the LFRR-ZU sector, that is a portion of the airspace of Brest FIR. In RESMI 6 airways merge, coming from the continental Europe and the Atlantic Ocean. After RESMI, aircraft trajectories can proceed through 8 different airways overflying Europe(see figure 1). The actual traffic needs to be sequenced to avoid conflicts.

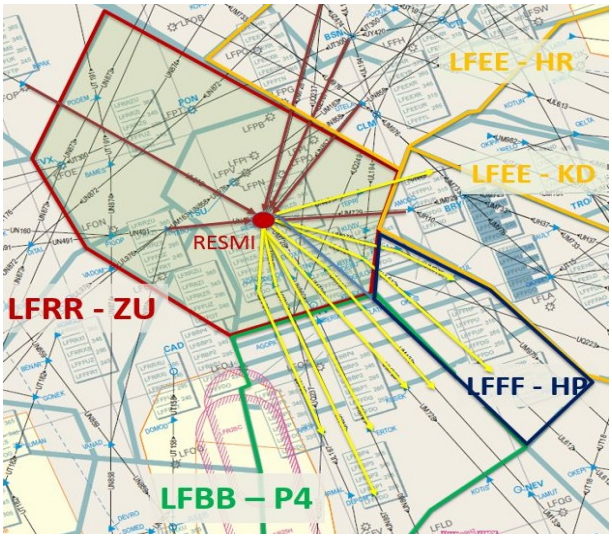

Fig. 1. Actual air traffic situation in Brest airspace: 6 incoming airways and 8 outgoing airways affect RESMI way-point

The final purpose is to have a possibility for the Air Traffic Controllers to give clearance to aircraft to directly cross the

airspace. We want to propose an advanced automation while maintaining the controller behind the final decision.

B. State of the art

Studies concerning an improved interaction between automatic systems and controllers have already been undertaken, as evidenced by ERASMUS [1]. This project proposes an air-ground cooperative work aiming at defining and validating innovative automation and concepts of operations for the En-route phase. Its goal is to propose an advanced automation while maintaining the controllers in the decision loop. In this project, as possible solution for conflicts, it has already been proposed an algorithm that can solve conflicts by minor alterations of the vertical/horizontal speeds or rate of climb/descent for a given aircraft. Some other studies, as the ones presented in [2], have used an automated optimization to resolve conflicts and to maintain separation between aircraft considering the possibility to control and modify the speed of an aircraft.

As regards our proposal to offer aircraft a possible direct route inside an airspace, a similar approach is already proposed in the actual air traffic. Some northern European Air Navigation Service Providers, like the Irish Aviation Authority and the UK National Traffic Services, already enabled airlines to fly direct routes during the night over Ireland into oceanic airspace. The airlines confirmed a predicted savings in terms of fuel and flight time and they request a further extension of this free route airspace. This was accommodated in cooperation with UK NATS by launching a new project called Night Time Fuel Saving Routes (NTFSR) across Ireland and UK airspace which allows direct routings to selected destinations during the night, resulting in further benefits [3]. Nevertheless, this implementation risks to affect air traffic controllers' workload, who are called to tackle new routes and, thus, new potential conflicts. That is why this direct route proposal should be addressed in an automatic way, like the approach we want to propose.

C. Given Data

The airspace sector we consider hosts trajectories that we have decided to represent through a network modeled as a graph $G = (P, E)$ in which aircraft are allowed to fly. P is the node set and E is the link set. We have $P = P_{in} \cup P_{out} \cup \{w\}$, where $P_{in} = \{1, ..., p_{in}\}$ represents the set of entry points, $P_{out} = \{1, ..., p_{out}\}$ represents the set of exit points and w is the considered crossing waypoint; p_{in} is the total number of entry points and p_{out} is the total number of exit points. The set E is composed of those links connecting every entry point to w and w to every exit point. Since one of our objectives is to give aircraft the possibility to straight cross the sector, we add to E all those links that directly connect every entry point with all the exit points.

Each trajectory is defined by a succession of nodes and links. In this network, we consider two types of trajectories: the direct trajectories and the classical trajectories. The first type (direct route) is composed by a succession of just three elements: $\{in, link, out\}$, while the second one is composed of five elements: $\{in, link_1, w, link_2, out\}$. Classical trajectories are those that are already planned by the ATM, direct trajectories are those that we want to suggest to ATC after analyzing the current traffic situation.

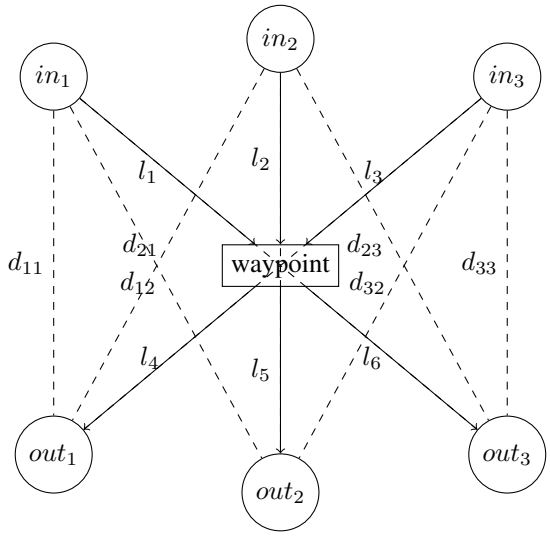

Fig. 2. Example graph representing a network of 3 entry points, a central way-point and 5 exit points. Direct routes are represented by a dashed line, classical routes by solid line.

In the graph presented in Figure 2, an example with three entry points and three exit points is shown. The node set is given by $P_{in} = \{in_1, in_2, in_3\}$, $P_{out} = \{out_1, out_2, out_3\}$ and $waypoint$. The link set is $E = \{l_1, l_2, l_3, l_4, l_5, l_6, d_{11}, d_{12}, d_{13}, d_{21}, d_{22}, d_{23}, d_{31}, d_{32}, d_{33}\}$, where l_i represent the links connecting the waypoint with the other points, while d_{ij} represent the links directly connecting an entry point i with an exit point j.

We then provide a list of flights in this network. For each flight $f \in F = \{1, ..., N_f\}$, the following data are given:

- identification number of the aircraft;
- traveled trajectory (we then know the entry point $p_{in}^{N_i} \in P_{in}$ and the exit point $p_{out}^{N_i} \in P_{out}$)
- entry time in the sector $t_{in}^{N_i}$ at entry point in_i
- entry speed $v_{in}^{N_i}$
- flight level FL

The fictitious network we will consider for our tests is formed by 5 aligned entry points, a central way-point and 5 aligned exit points, as shown in Figure 8.

D. Decision variables and constraints

We consider three possible maneuvers for each flight to solve a potential conflict:

- Modification of its entering speed in the sector through a speed decision variable $\Delta v_i \in [\Delta v_{min}^i, \Delta v_{max}^i]$. We assumed this interval to be discretized in speed-steps, according with practice of controllers, who modify aircraft speed by accelerating or reducing by a discretized value. In this study, for each flight we set $\Delta v_{min} = -40kts$ and $\Delta v_{max} = 20kts$;

- Change of its entering flight level in the sector through a FL decision variable $\Delta FL \in [\Delta FL_{min}, \Delta FL_{max}]$. In this study we consider $\Delta FL \in [-4; 4]$;
- Choice between the classical route through w and a direct route from the entry point $p_{in}^{N_i}$ to the exit point $p_{out}^{N_i}$ through a boolean value $D_i \in \{true, false\}$, where we consider *true* if the aircraft is allowed to follow a direct route, *false* otherwise.

Since we want to maximize the number of direct routes and minimize the deviations in speed and FL, the first situation we feed the simulator with, is the ideal situation where each aircraft f is assigned a direct route ($D = true$), no speed modification ($\Delta v = 0$) and no flight level change ($\Delta FL = 0$). All this information is contained in a decision variable vector **u**=(D, Δv, ΔFL) where, for each aircraft, are stored all the applied changes.

E. Separation rules

Two separation requirements have to be satisfied:

- **Spatial separation constraint**: in the sector we consider, air traffic regulations impose an horizontal minimum separation of 5 nautical miles and a vertical minimum separation of 1000 feet. One can consider that at any given time, each aircraft has a bounded and closed reserved block of airspace defined by a three-dimensional cylinder in which other aircraft are not allowed to enter;
- **Time separation constraint**: proposing possible direct routes, new intersection points between trajectories are created. Each of these points can be overflown by an aircraft at most every minute.

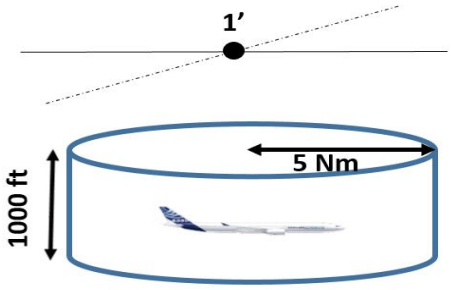

Fig. 3. Separation Rules: a) Intersection points can be overflown at most every minute; b) An aircraft has a reserved block of airspace corresponding to a cylinder of 5NM radius and 1000 feet height.

F. Conflict detection

Before describing the solution used to detect conflicts, we present the assumptions considered in our study:

- The airspace is considered as an euclidean space: latitudes and longitudes on the Earth surface are projected into (x, y) coordinates, while the altitude is represented by the z coordinate;
- Each trajectory $i = 1, ..., N$ is discretized through a set of 4D points (x, y, z, t) sampled with a given sampling time t_s. In our study we consider $t_s = 10s$;
- The entire network is built in the first quadrant of the Cartesian system in order to have only positive coordinates and thus simplify the calculations.

As already mentioned in SectionI-B, to detect conflicts we use the approach used in [4]. We proceed as described below and we consider as example the trajectories A, B and C in Figure 4.

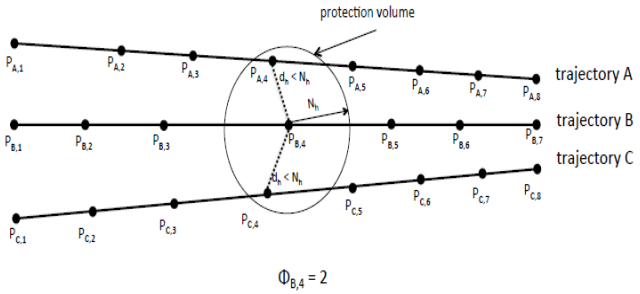

Fig. 4. Interactions, $\Phi_{B,4}$, at sampling point $P_{B,4}$ of trajectory B.

We define an *interaction at a trajectory point* $P_{i,k}(u_i)$ to be the sum of all the conflicts associated with point $P_{i,k}(u_i)$, where u_i is the i^{th} component of u. We further define the *interaction,* Φ_i*, associated with trajectory* i, as: $\Phi_i(u) := \sum_{k=1}^{K_i} \Phi_{i,k}(u)$ where K_i is the number of trajectory points obtained through the discretization, mentioned in the assumptions, of the trajectory of the i^{th} flight. Figure 4 illustrates the case of trajectory $i = B$ at the trajectory point $P_{B,4}$. Finally, *interaction between trajectories*, Φ_{tot}, for a whole traffic situation is simply defined as:

$$\Phi_{tot}(u) := \sum_{i=1}^{N} \Phi_i(u) = \sum_{i=1}^{N} \sum_{k=1}^{K_i} \Phi_{i,k}(u). \qquad (1)$$

To check if any conflict occurs, one needs to compute the interaction, Φ_{tot}, between the N aircraft trajectories. To avoid the $\frac{N(N-1)}{2}$ time-consuming pair-wise comparisons, we propose a 4D grid-based conflict detection scheme as illustrated in Figure 5 (see [4], [5] for further details). First, we define a four-dimensional (3D space + time) grid. Its spatial dimension has to be large enough to include the considered airspace and the time-span has to be long enough to cover the period considered. The size of each cell in the x, y, and z directions is defined by the minimum separation requirements, $N_h = 5$ NM and $N_v = 1,000$ feet. The size of the cell in the time domain is set according to the given discretization step size, t_s. To detect conflicts, every t_s seconds, only the surrounding cells will be checked to verify the separation constraints. In fact, the violation of the protection volume can only occur when the considered points are in the same or in the adjacent grid cells.

G. Objective function

The main objective is to guarantee a conflict-free scenario when direct routes are proposed to aircraft. We face then a minimization problem where we have to bring to zero the

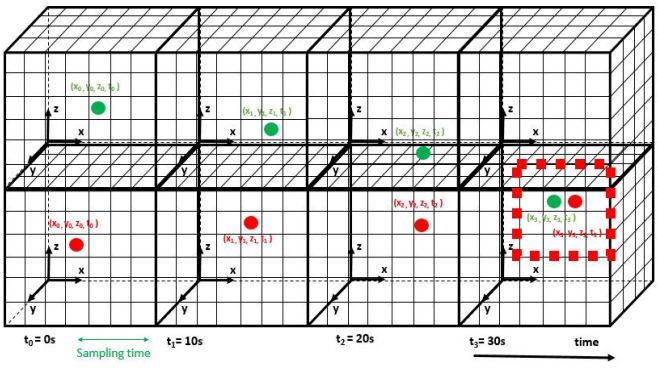

Fig. 5. 4D Conflict Grid used to detect potential conflicts

number of conflicts C. At the same time, since aircraft are also requested to change either flight level or speed to avoid potential conflicts, we want to minimize as well the changes of flight level ΔFL and the changes of speed Δv requested. The objective function to be minimized is then

$$F(\mathbf{u}) = \mu_1 C(\mathbf{u}) + \mu_2 \sum_{i=1}^{N_f} |\Delta v(\mathbf{u})| + \sum_{i=1}^{N_f} \mu_3 \, |\Delta FL(\mathbf{u})|$$

where μ_1, μ_2 and μ_3 are weighting coefficient that can be tuned to give the desired priority to the three parameters. μ_1 shall have the greatest value since the main objective is to set at zero the number of conflicts. To maximize the number of direct routes we act as described in Section I-D: we start the simulation process with the ideal situation where each aircraft follows a direct route.

II. Optimization Algorithm

A. Simulated Annealing

The optimization approach relies on a standard Simulated Annealing (SA) algorithm. SA is a metaheuristic optimization method introduced by S. Kirkpatrick in 1983 and V. Cerny in 1985 [6]. Its popularity comes from its ability to avoid getting trapped in local minima and to find a near-global optimal solution for NP-hard combinatorial optimization problems [7], [4]. This method is based on an analogy with the physical phenomenon experienced in metallurgy where the state of a material is modified by controlling its temperature. The first phase consists in heating up the material in order to reach the point of fusion characterized by a high-energy state. Then, following a cooling schedule, the solid is brought to solidification, characterized by a low-energy state. It is important to choose an appropriate duration for each step during the cooling in order to allow the material to reach its thermal equilibrium before further reduction of temperature. In this way, it is possible to reach the solid state with minimal energy. If the temperature is decreased too rapidly, the solid enters a metastable state of non-minimal energy, remaining stuck in a locally-optimal energy level. In our optimization problem, the cost function to be minimized is equivalent to the energy of the physical problem. A control parameter that decreases as the number of iterations grows, plays the role of the temperature schedule. Finally, a determined number of iterations at each temperature step plays the role of the time duration the material is kept at each temperature stage.

The Metropolis Algorithm, developed in 1953, simulates the physical process of annealing on a computer. Considering a current state i of energy E_i, the algorithm creates a disturbance in order to transform the current state in a new one. If the energy E_j of the new state j is smaller than E_i, the new state is accepted as the new current state. If E_j is greater than E_i, state j is accepted as the new state with a probability $P = e^{(E_j - E_i)/(k_b T)}$, where T is the temperature and k_b is Boltzmann's constant [5]. At higher temperatures there are more possibilities of acceptance of a higher energy state. The basic idea is that, in a first moment, a solution that increases the energy of the current state is occasionally accepted, in order to escape from local minima; in a second moment, when the temperature decreases, such possibility is less and less frequent. Other optimization methods can guarantee convergence to a global optimum to the problem. However, metaheuristic methods have been shown to provide good solutions to the air traffic management problem within reasonable computational time [4].

Algorithm 1 Simulated Annealing

Require: $\vec{x_i}$, C_0, L_0, $k = 0$
repeat
  for $l = 0 \rightarrow L_k$ **do**
    Generate a solution $\vec{x_j}$ from the neighborhood $S_{\vec{x_i}}$ of the current solution $\vec{x_i}$
    If $f(\vec{x_j}) < f(\vec{x_i})$, then $\vec{x_j}$ becomes the current solution; otherwise, $\vec{x_j}$ becomes the current solution with probability $p = e^{\frac{f(\vec{x_i}) - f(\vec{x_j})}{C_k}}$
  end for
  $k = k + 1$
  calculate (L_k, C_k)
until $C_k \simeq 0$

B. Objective Evaluation

The objective function is evaluated thanks to a computer simulation process which requires a simulation environment. The optimization algorithm controls the vector of decision variables, X, which are used by the simulation process in order to compute the performance (quality), y, of such decisions, as shown in Figure 6.

In the standard simulated annealing algorithm, a copy of a state space point is requested for each proposed transition. In fact, a point $\vec{X}_j$ is generated from the current point $\vec{X}_i$ through a copy in the memory of the computer. In our study, the simple process of implementing such a copy would be inefficient and would reduce the performance of simulated annealing. That is why we consider a much more efficient operator called *come back*, which cancels the effect of a generation. Let G be the generation operator which transforms a point from $\vec{X}_i$ to $\vec{X}_j$:

$$\vec{X}_i \xrightarrow{G} \vec{X}_j$$

the comeback operator is the inverse G^{-1} of the generation operator.

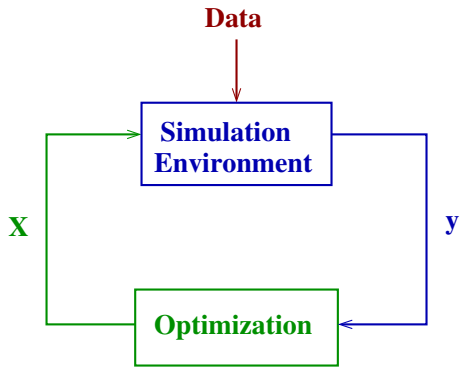

Fig. 6. Objective function evaluation based on a simulation process

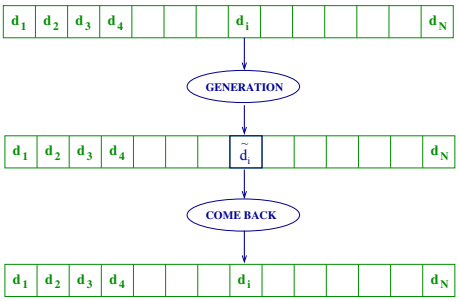

Fig. 7. Optimization of the generation process. In this figure, the state space is built with a decision vector for which the generation process consist of changing only one decision (d_i) in the current solution. If this modification is not accepted, this component of the solution recovers its former value. The only information to be stored is the integer i and the real number d_i.

Such a generation modifies only one component of the current solution. The vector $\vec{X}_i$ can then be modified without being duplicated. Depending on the value obtained when evaluating this new point, two options may be considered:

1) the new solution is accepted and, in this case, only the current objective function value is updated.
2) else, the come back operator G^{-1} is applied to the new position in order to come back to the previous solution, again without any duplication in the memory.

This process is summarized in Figure 7.

III. Implementation

This section presents how the problem has been implemented through an algorithm developed in Java, describing all the steps that are executed when the program is run.

A. Pre-processing

First of all, the node list and the route list are automatically read and the corresponding points and links are built in the space. Afterwards, a list of flights belonging to this network is read and processed. At this stage, every external information has been read and translated according to the mathematical model presented in Section I-C. The simulation process is then launched.

B. Simulation and optimization process

The SA **heating up loop** is launched and its output provides us with the initial temperature T_{init} we need to start the cooling loop. This temperature is computed by first generating 100 deteriorating transformations at random and then by evaluating the average variations, $\Delta\Phi_{avg}$, of the objective function values. The initial temperature, c_0, is then deduced from the relation: $c_0 = e^{\frac{\Delta\Phi_{avg}}{T_0}}$, where τ_0 is the initial acceptance rate of degrading solutions (which it is empirically set).

The **cooling loop** can now start. In order to reach an equilibrium, a sufficient number of iterations, denoted L_k, have to be performed at each temperature step k. In our case, we assume for simplicity purposes that L_k is constant and empirically set.

We start by initializing the *state* as presented in Section I-D: each flight has a decision vector $u = (true, 0, 0)$. In the *state* we store all the information regarding the decision for each aircraft. Then, we compute for each flight the 4D points representing its trajectory. To optimize the memory allocation, these 4D points are not directly placed in the 4D grid. Their coordinates are stored in a specific array along with two indexes used to identify them: a first index i referring to the trajectory to which they belong, and a second index j that refers to the sequence number inside that trajectory. For example, the point $P_{5,34}$ will be the 34^{th} point of trajectory 5. Thanks to these only 2 indexes we are able to refer to any 4D Point and we are able to retrieve in every moment its spatial and time coordinates. This allows us to save memory space. We refer to this way to indicate the points as *4DPointInfo*. The 4D grid is then populated with the *4DPointInfo*.

At this stage, potential conflicts have to be detected following the procedure presented in Section I-F. If two or more points are detected to be in the same cell or in adjacent ones, their distance have to be computed to check if the protection volume is violated. To do that, their spatial coordinates are retrieved from the above-mentioned array. Finally, the objective function is computed.

From this point, a *for* loop is triggered (Algorithm 2):

Algorithm 2 Cooling

```
for (i = 0; i < nbTransitions; i++) do
  x_j = generateNeighbour(x_i);
  computeFunction y_j = f(x_j)
  if accepted then
    x_i = x_j
    y_i = y_j
  else
    comeBack
  end if
end for
```

First, a *neighborhood function* is defined to generate a local change from the current solution. One of the three decision parameters of one of the N_f flights is randomly changed. Simultaneously, the *4DPointInfo* of that flight are removed

from the 4D grid, the 4D points are re-computed and its new *4DPointInfo* are placed in the 4D grid. To generate a neighborhood solution, instead of simply choosing randomly a flight f in the active-flight set, we use a method similar to the *roulette wheel* selection, as presented and used in [2]. This way of operating augments the likelihood that a flight involving many conflicts, or its neighboring aircraft, will be chosen.

Once more, all the potential conflicts are detected and the objective function is anew calculated. If the neighbour is accepted, it becomes the new current solution and its objective value becomes the new value for the objective function. If it is not accepted, all the 4D points that had been computed and all the *4DPointInfo* that had been placed in the grid, have to be removed and replaced with the previous ones. The temperature is then decreased following a geometrical law, $c_{k+1} = \alpha c_k$, where $0 \leq \alpha \leq 1$ is a pre-defined constant value. The process is then restarted. This cooling loop goes on either until the best solution is found, i.e. $F(\mathbf{u}) = 0$, or until a target temperature set up by the user is reached. In our study this temperature is set to be $T_{stop} = 0.001\, T_{init}$.

Conflicts computation: In the SA optimization process, the computation of the objective function is repeated many times, according to the number of iterations set by the user. Therefore it must be computed as efficiently as possible. To avoid checking interactions over all the N trajectories at each iteration, we proceed as follows. First, the 4D grid is initialized with every cell empty. Then, the initial N trajectories, corresponding to the initial value of the decision vector, u (with all its component = $[true, 0, 0]$), are placed in the 4D grid and the *current* interaction, Φ_{i_C}, associated with each trajectory i, and the current total interaction between trajectories, Φ_{tot_C}, are computed.

During the optimization process, at each iteration, only one parameter can be changed. This implies that only one of the N trajectories is changed. We denote it as τ. To update the value of total interaction, we first remove τ from the 4D grid. Therefore, the interaction associated to τ is set to an intermediate value $\Phi_{\tau,inter}(u) = 0$. It should be noted that the interaction measurement is symmetrical: if $\Phi^{ij}(u)$ denotes the *contribution of trajectory i to the interaction associated with trajectory j*, then $\Phi^{ij}(u) = \Phi^{ji}(u)$. Let $\mathcal{N}_i$ be a set of trajectories currently interacting with trajectory τ. The interaction associated with trajectory $j \in \mathcal{N}_i$ over τ, is set to an intermediate value $\Phi_{j,inter}(u) = \Phi_j(u) - \Phi_\tau$. Thereafter, the *modified* trajectory corresponding to the new decision variable values, is placed in the 4D grid and the interaction detection is performed only over it. Then, the new interaction Φ_τ is computed. Again, the interaction associated with each trajectory, j, interacting with τ is updated as follows: $\Phi_j(u) = \Phi_{j,inter}(u) + \Phi_\tau$. Finally, the total interaction between trajectories is simply computed as $\Phi_{tot}(u) = \sum_{i=1}^{N} \Phi_i(u)$. This interaction computation method allows us to update the value of the objective function when one trajectory is modified within a very short computation time, since we do not need to compute the change of interaction for decisions that are not modified at the current optimization iteration.

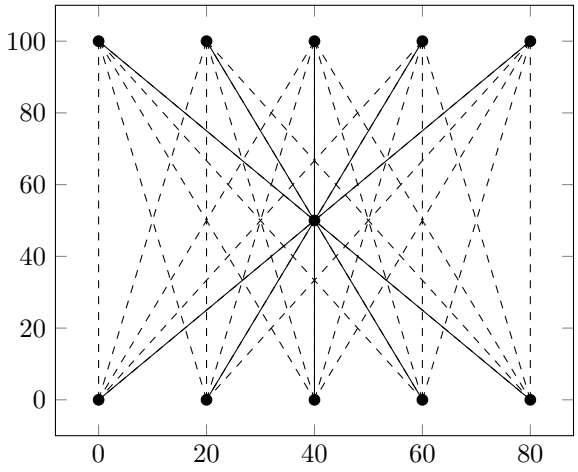

Fig. 8. Network used for the simulation: 5 aligned entry points and 5 aligned exit points symmetrically located with respect to the central way-point.

IV. Results

A. Environment and Parameter Values

The proposed methodology is tested with the fictitious network shown in Figure 8. Its nodes are represented by a central way-point, 5 entry points aligned along an entry line, and 5 exit points aligned along an exit line. A distance of 20 NM separates each entry point as well as for each exit point. The exit line is 100 NM distant from the entry line. This results in an airspace whose total dimension is 100x80 NM. Trajectories are represented by a solid line if they overfly the way-point or by a dashed line if they are direct routes. All the flights following these trajectories are read from a randomly generated list using the parameter presented in Table I. Looking at entry time set of possible values, we can see that we are considering a time frame of 2 hours and 46 minutes (10000s).

The parameter values chosen to specify the instance considered, and the empirically set parameters defining the overall SA problem-solving methodology, are given in Table II. In our study, the conflict weight coefficient μ_1 is set to a much higher value than μ_2 and μ_3 in order to ensure that the conflict resolution is the first priority. The SA adapted to solve the strategic trajectory planning problem is implemented in Java. We address this problem instance with an AMD Radeon 2 GHz processor with 6 Gb RAM.

B. Simulation Results

Numerical results obtained from several simulations are reported in Table III. Twelve different simulations have been run on 8 different scenarios, increasing from time to time the number of aircraft fed to the network. In those scenarios where a conflict free solution has not been found, more simulations

Entry parameter	Possible values
Entry time	$0 < t_{in} < 10000s$
Entry speed	$400 < v < 420kts$
Entry flight level	$260 < FL < 320$

TABLE I
POSSIBLE VALUES FOR THE ENTRY PARAMETERS FOR THE RANDOMLY GENERATED FLIGHTS

Parameter	Value
Number of iterations at each temperature step	100
Geometrical temperature-reduction coefficient α	0.95
Final temperature T_{stop}	$0.001\,T_{init}$
Conflict weighting coefficient μ_1	100
FL weighting coefficient μ_2	1
Speed weighting coefficient μ_3	1
Discretization step t_s	$10s$
Speed deviation Δv in kts	$-40 < \Delta v < 20$
FL deviation ΔFL	$-4 < \Delta$ FL< 4

TABLE II
EMPIRICALLY-SET (USER-DEFINED) PARAMETER VALUES

have been run changing randomly the entry values of the aircraft, in order to check if it was possible to have 0 residual conflicts.

A free-conflict scenario has been reached for scenario A, B, C, D and E. This shows that the considered network is suitable for up to about 210 aircraft. For a heavier traffic condition, there remain unsolved conflicts. It is recalled that we are considering a time frame of about 2 hours and 45 minutes and a fictitious airspace of 100x80 NM.

In Figure 9, 10 and 11 it is possible to observe the evolution of the number of conflicts, the total ΔFL and the total Δv respectively. In those scenarios where more tests have been conducted (namely Scenario F and G), average values of the three different tests have been considered for conflicts, ΔFL and Δv. It is possible to notice that, whatever the scenario, only few iterations are needed to drastically decrease the number of potential conflicts. For those scenarios that reach a conflict-free situation, less than 14 iterations are required to solve any conflict. As regards the total ΔFL and Δv, both present the same trend for each scenario: to give priority to the conflict resolution, their values show an initial increase. Subsequently, when the number of conflicts is remarkably reduced, they also start to decrease so that the objective function continues to improve (see Figure 12). Both Figure 9 and 12 have been zoomed in the first iterations (compared with the others figures) to see clearly the decreasing phase.

Sim	Scenario: No A/C	Initial conflicts	Residual conflicts	No direct route	Modified flights
1	A: 50	0	0	37	15
2	B: 100	115	0	58	48
3	C: 150	1554	0	63	101
4	D: 200	5434	0	71	144
5	E: 210	6041	0	63	167
6	F: 225	12338	4	63	181
7		6141	4	64	176
8		9402	6	62	177
9	G: 250	13005	8	67	204
10		13661	8	66	204
11		15811	5	79	199
12	H: 300	26633	19	79	248

TABLE III
SIMULATION RESULTS

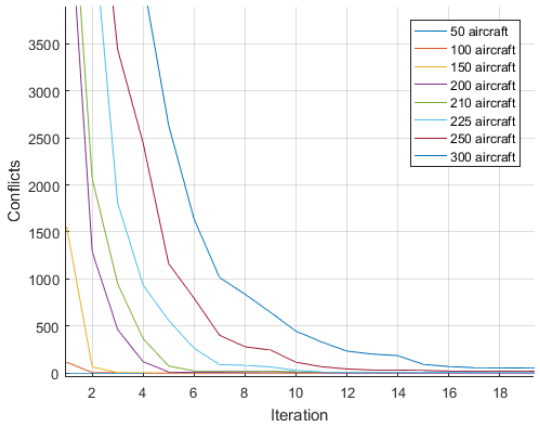

Fig. 9. Number of conflicts evolution with respect to iterations

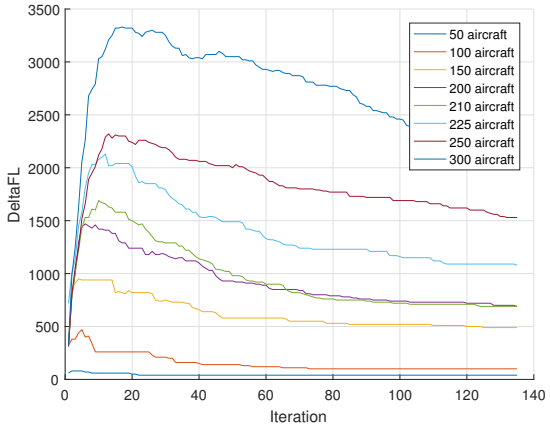

Fig. 10. DeltaFL evolution with respect to iterations

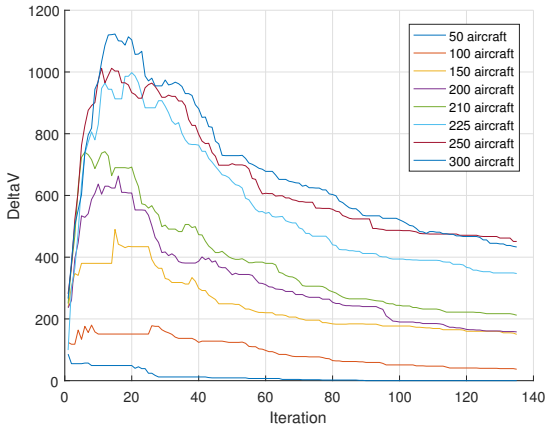

Fig. 11. DeltaV evolution with respect to iterations

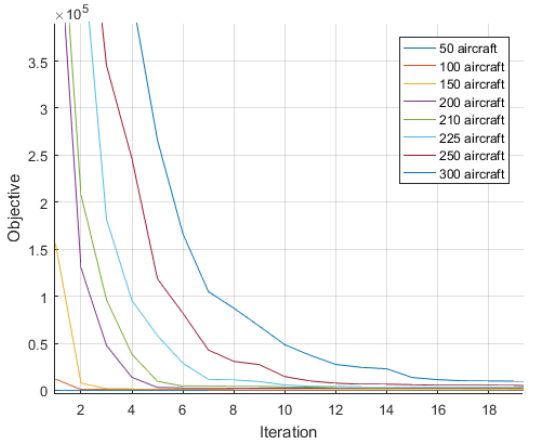

Fig. 12. Objective function evolution with respect to iterations

The percentage of flights that are subjected to a modification on the trajectory compared to the initial situation (where each aircraft follows a direct route, with no ΔFL and no Δv) is presented in Figure 13 for each scenario. Finally, in Figure 14 it is possible to observe the percentage of direct routes at the end of the optimization process. The more the network is busy, the less direct in term of percentage.

All the simulations that led to a conflict-free situation have been run in a maximum computation time of 14 minutes. That means that 6.2 seconds are needed for each one of the 135 iterations requested to reach T_{stop}. Since after only 14 iterations we have been able to find a conflict-free situation, we can say that only 90 seconds are needed to reach a first possible solution. The remaining time is used to improve and refine the solution.

V. Conclusion and Future Development

The main purpose of this paper is to study the feasibility of giving clearance to aircraft to directly cross the airspace while maintaining a conflict-free situation. Considering a time frame of 2 hours and 46 minutes in a network of 100x80 NM, the

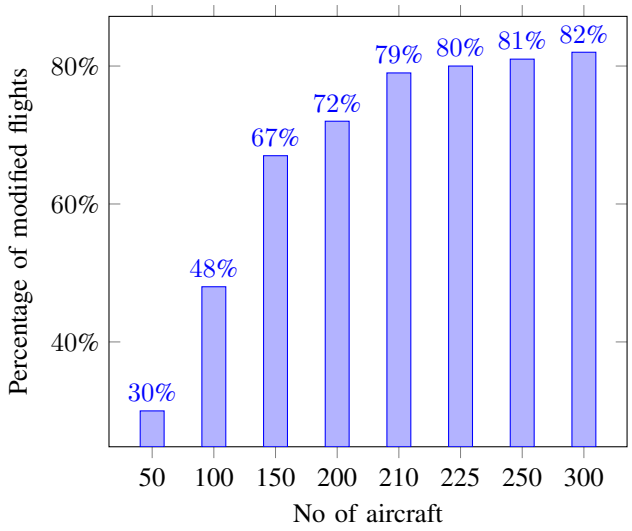

Fig. 13. Percentage of modified flights

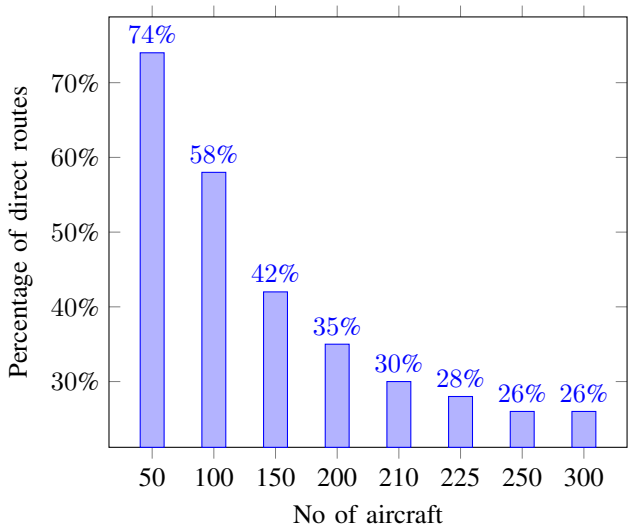

Fig. 14. Percentage of direct routes

algorithm successfully reached a conflict-free scenario for up to 210 aircraft. Up to 71 direct trajectories has been permitted for a scenario with 200 aircraft, that corresponds to a 35% of direct routes. For a low traffic situation it has been possible to achieve up to a 74% of direct routes.

Thanks to these simulations, the algorithm has been tested and verified. Positive results have been found and then the algorithm is now ready to be used to solve the real case presented in Section I-A(RESMI crossing point in Brest ACC (France)). We want to highlight that if the proposed strategy is applied to several airspace sectors along the entire route of an aircraft, this could lead to a significant flight time reduction and to a decrease on both fuel consumption and CO_2 emission.

Further developments may focus on more complex networks where multiple way-points are considered. New parameters could be introduced to allow a new possibility of direct

route aiming at the exit point starting not only from the entry point, but from any point along the path inside the considered airspace. This would allow to apply our strategy to bigger airspace. An example with two way-points and its new parameters α and β is shown is Figure 15.

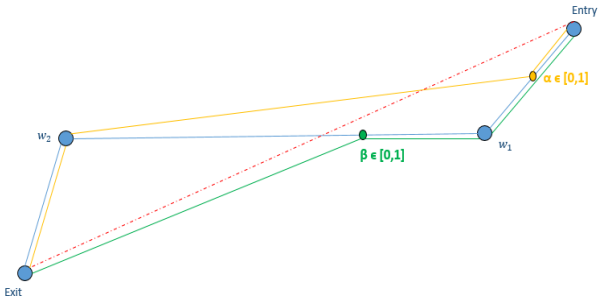

Fig. 15. Direct route proposal from any point of a trajectory overflying two way-points. α and β are used to denote any position along the different stretches of the trajectory

REFERENCES

[1] Marc Brochard, ERASMUS - En Route Air Traffic Soft Management Ultimate System. Eurocontrol Experimental Centre, Bretigny, France. 2005.

[2] Ji Ma, D. Delahaye, M. Sbihi, M. Mongeau. Merging Flows in Terminal Maneuvering Area using Time Decomposition Approach. 7th International Conference on Research in Air Transportation (ICRAT 2016), June 2016, Philadelphie, PA, United States.

[3] CANSO, Boeing. Accelerating Air Traffic Management Efficiency: A Call to Industry. Pag 24-25. February 2012.

[4] Supatcha Chaimatanan. Strategic planning of aircraft trajectories. Optimization and Control [math.OC]. Universite Paul Sabatier - Toulouse III, 2014. English. tel-01064452

[5] D. Delahaye and S. Puechmorel. Modeling and Optimization of Air Traffic. Wiley-ISTE, 2013.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. Science, 220:671–680, 1983.

[7] Supatcha Chaimatanan, Daniel Delahaye, Marcel Mongeau. A methodology for strategic planning of aircraft trajectories using simulated annealing. ISIATM 2012, 1st International Conference on Interdisciplinary Science for Air traffic Management, Jun 2012, Daytona Beach, United States. 2012. hal-00912772

[8] EUROCONTROL. European Free Route Airspace Developments. Edition 1.0. March 2015. http://www.eurocontrol.int/articles/free-route-airspace