Does ATM Need Centralized Coordination?
Autonomous Conflict Resolution Analysis in a Constrained Speed Environment
Nicolas Durand, Nicolas Barnier

To cite this version:
Nicolas Durand, Nicolas Barnier. Does ATM Need Centralized Coordination? Autonomous Conflict Resolution Analysis in a Constrained Speed Environment. ATM seminar 2015, 11th USA/EUROPE Air Traffic Management R D Seminar, Jun 2015, Lisboa, Portugal. <hal-01168655>

HAL Id: hal-01168655
https://hal-enac.archives-ouvertes.fr/hal-01168655
Submitted on 26 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Does ATM Need Centralized Coordination?  
Autonomous Conflict Resolution Analysis in a Constrained Speed Environment

Nicolas Durand and Nicolas Barnier  
ENAC/MAIAA  
31 055 Toulouse, France  
firstname.surname@enac.fr

Abstract—The concept of Free-Flight, introduced in the 90s, opened a debate on the efficiency of letting aircraft deal with conflicts without any centralized control. Many models have been proposed for autonomous aircraft solvers but their efficiency is not well-known. In this paper, we experiment powerful algorithm derived from robotics which is able to deal with thousands of robots in very small spaces, and show how its performance plummets when speeds are constrained. We also compare this autonomous algorithm with a centralized approach using evolutionary computation on a complex example to point out their relative performance in a speed constrained environment. This comparison provides scientific arguments for the necessity of centralized air traffic control.

Keywords: conflict resolution, free-flight, geometrical algorithms, evolutionary computation

INTRODUCTION

When the concept of Free-flight emerged in the 90s, a large debate started on the possibility of eliminating centralized control systems by developing autonomous algorithms for solving conflicts. The first effective approach used sliding forces to coordinate maneuvers between aircraft [1]. Potential or vortex fields [2] as well as a model based on an analogy with electrical particle repulsion [3] were also used.

However, these approaches were never compared to a centralized method and the behaviors of the algorithms were not tested in very dense environments. They also needed to be tuned carefully to be effective and finding general rules to do so did not seem possible.

In 2001, we proposed a token allocation strategy combined with an A* algorithm to solve conflicts with realistic maneuvers [4], [5]. Even if some maneuvers could be simultaneously decided, a complete ranking of aircraft was necessary and finding an optimal ranking has been shown to be problem dependent [6]. We also tried neural networks on the two-aircraft problem [7] but they could not be generalized to handle more aircraft.

Geometrical approaches have also been widely studied [8]–[11]. The powerful technique developed by Van den Berg et al. [11] can handle thousands of agents in a small space. It was applied to aircraft by Snape et al. [10], but the hypotheses of the algorithm require simultaneous vertical and horizontal speed changes, which are not realistic.

In the 90s, we showed that conflict resolution was highly combinatorial and could not be globally solved with local optimization methods [12]. We proposed a genetic algorithm to solve multiple aircraft conflicts using a centralized approach with simple maneuvers (similar to those used by air traffic controllers) and showed that we were able to solve every conflict on real traffic data [13].

Centralized methods can be divided into two main categories. Some methods [14]–[15] use greedy sequential algorithms to optimize trajectories one by one after ranking the aircraft. However, finding an appropriate ordering is challenging [6]. The others, mentioned in the next paragraph, try to find the global optimum without the need to prioritize aircraft.

Using evolutionary computation, our team was the first to address conflict resolution globally [12]. Others later introduced a powerful Semidefinite Programming approach [17], [18], which is also able to handle multiple aircraft in a single scenario. However the solution is only locally optimal and the model requires constant speeds and perfect trajectory prediction. In the early 2000s, a mathematical model using Mixed Integer Linear Programming, which could be solved by CPLEX and ensured the global optimality of the solution, was proposed [19], [20]. The model was extended in 3D in 2008, but required constant speed during climbing phases [21]. Uncertainties on the trajectory heading were added to the horizontal model in 2009, but all maneuvers still needed to be executed at the same time (at every optimization step) [22]. The method is powerful but cannot be used for developing a realistic advisory tool for controllers. Finally in 2013, our team used a Constraint Programming approach to globally solve complex conflicts [23]. For each aircraft, a number of alternative trajectories and a matrix of pairwise conflicts were precalculated, taking various uncertainties into account, before the optimization process was performed.

In this study, we use the autonomous approach proposed by Van den Berg et al. in [11]. We relax some realistic hypotheses on trajectories and revert to a simple horizontal model in order to assess the efficiency of an autonomous conflict resolution solver for aircraft with various levels of maneuverability in the horizontal plane. In section II of this article, we detail the geometrical approach developed by Van den Berg et al. [11]. Then we test it in a dense environment in section II and...
constrain the aircraft speeds and turning angles to show how the performance of the algorithm changes. In section III we compare the autonomous algorithm to a centralized approach based on an Evolutionary Algorithm and use an example to show the advantage of coordinating maneuvers when the speed is constrained. The last section discusses the consequences of speed constraints on the autonomous solver and highlights directions for future work.

I. GEOMETRICAL MODEL

This section describes the algorithm developed in [11]. Let $d$ be the standard separation between aircraft and $\tau$ be the lookahead time. In figure 1 let us consider two aircraft A and B. We can represent the position of aircraft B in the referential of aircraft A. If we draw a circle of radius $d$ around aircraft B, the two lines issued from position A, tangent to the circle of radius $d$ form a cone. If the relative speed $\vec{r}$ lies in this cone, a conflict will occur in the future. If we draw a circle of size $\frac{d}{2}$ tangent to the two previous lines, we obtain a new zone bounded by the bold line in figure 1. It is then straightforward to understand that a conflict will occur within time $\tau$ if and only if $\vec{r}$ lies in this zone.

When a potential conflict occurs (i.e. $\vec{r}$ is in the forbidden zone), $\vec{r}$ is projected on the nearest bold line to escape the forbidden zone (see figure 2) and solve the conflict pairwise between aircraft A and B. The necessary speed change $\vec{s}_c$ to move $\vec{r}$ out of the forbidden zone is then divided in two and shared by both aircraft.

We can then define two half-planes limited by lines perpendicular to $+\vec{r}$ and $-\vec{r}$: if the new values of $\vec{v}_A$ and $\vec{v}_B$ are chosen in their respective half-plane, then the new relative speed $\vec{r}$ is out of the forbidden speed zone. An example of the modified speeds is shown in figure 3.

In order to take into account more than two aircraft, the model can be extended. For each pair of aircraft, a half-plane is created. In figure 4, the intersection of the half-planes defined by aircraft B, C and D using the previous principle represents the acceptable speed zone for aircraft A.

Because the possible maneuvers of an aircraft are heavily constrained by aerodynamic laws and operations rules, the speed vector change at each time step is limited in our model. For the sake of simplicity, we can choose to change the aircraft speed in a box of length $2s_{lon}$ and of width $2s_{lat}$ (see figure 5). The intersection of this box and the previous area defines a permitted speed zone for aircraft A. If we project the optimal speed (heading to the destination) on this zone we can define the new speed for aircraft A.

If the permitted speed zone is empty, we uniformly increase the acceptable speed zone until the intersection with the speed change box is not empty. This new speed, although it no longer
Fig. 4. Acceptable speed zone for aircraft A.

Fig. 5. New speed for aircraft A in the permitted speed zone (dashed).

Fig. 6. A scenario example with 10 aircraft.

Fig. 7. A scenario example with 100 aircraft.

II. EXPERIMENTAL RESULTS WITH VARIOUS SPEED RANGES

The previous algorithm has proved to be very efficient in robotics [11]. It is also successfully used to mimic crowd
guarantees zero conflict for $\tau$ seconds, represents a satisfactory solution, i.e. with the least conflicts.

The same algorithm can be extended to avoid obstacles such as sector edges and restricted (e.g. military) zones. In this particular case, the speed change is not divided but totally allocated to the aircraft.
movements or building evacuations. It is easy to parallelize and can thus handle thousands of objects at the same time.

However, we need to assess its behavior in a speed constrained environment to check if it could be applied to air traffic conflict resolution. Therefore, we test it in a $500 \times 500$ nautical mile sector with varying numbers of aircraft $n$ and different parameters for $s_{\text{lat}}$ and $s_{\text{lon}}$. We only focus on horizontal maneuvers in a single flight level. In the following, low density is defined as 10, 20, 30 aircraft, medium as 50 aircraft, and high as 100, 120 or 130 aircraft. Figure 6 and 7 show the resulting radar tracks for two instances with 10 and 100 aircraft respectively.

A. Influence of the speed constraint $s_{\text{lon}}$

The aircraft speeds are set to 1 (speeds are normalized), which means that we first set all aircraft to fly at the same speed. This is not totally realistic, but similar results are obtained with nominal speeds varying by $\pm 20\%$. The time step is set to 3 seconds. For a standard turning rate of 3 degrees per second we round the value of $s_{\text{lat}}$ at 0.05. We fix $\tau = 360$ s (i.e. 6 min) because it is a good compromise: if the lookahead time is too long, the algorithm is less efficient for high densities because too many conflicts are simultaneously taken into account and the permitted zones often are empty; whereas if $\tau$ is too short, some conflicts appear too late to be resolved.

We first run 100 executions for different values of $n$ and $s_{\text{lon}}$ and for $s_{\text{lat}} = 0.05$. For each execution, the origins and destinations of the aircraft are randomly chosen on the edges of the square. We make sure that aircraft are separated by at least 2 separation standards (10NM) at the origin and the destination. A quarter of the traffic goes from the east side to the west side, another quarter from west to east, a quarter from north to south and the last one from south to north.

For each scenario ($n$, $s_{\text{lon}}$), we first count the number of runs which fail because aircraft are “pushed” out of the $500 \times 500$ nautical mile sector before reaching their destination. We then count the number of runs which do not solve every conflict. For the remaining runs, we measure the trajectory lengthening (the ratio of the modified trajectory and the original trajectory).

Table I gives the experimental results for densities varying from 10 to 50 aircraft and various $s_{\text{lon}}$ values representing the speed range. E.g. $s_{\text{lon}} = 0.01$ means that the updated aircraft speed can be chosen within $\pm 1\%$ of the current speed, whereas with $s_{\text{lon}} = 1.$, the new speed can be within $\pm 100\%$ of the former one. The latter is totally unrealistic for aircraft but is tested to compare the efficiency of the algorithm for different speed ranges. Table II gives results for higher densities varying from 100 to 130 aircraft and various $s_{\text{lon}}$ values.

When $s_{\text{lat}}$ becomes very small, aircraft can only avoid each other by changing headings. For converging aircraft with close headings, aircraft tend to choose parallel routes and postpone conflicts instead of solving them. This is an issue because aircraft slowly get away from their original tracks and reach the edge of the sector without solving conflicts. When such a border violation occurs, the instance is considered a failure.

<table>
<thead>
<tr>
<th>nb acft</th>
<th>$s_{\text{lon}}$</th>
<th>Tot Fail</th>
<th>Border Viols</th>
<th>Rem Conf</th>
<th>mean delay</th>
<th>max delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0001</td>
<td>46</td>
<td>44</td>
<td>2</td>
<td>1.006994</td>
<td>1.078254</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>46</td>
<td>43</td>
<td>3</td>
<td>1.005470</td>
<td>1.132507</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>29</td>
<td>28</td>
<td>1</td>
<td>1.004209</td>
<td>1.025716</td>
</tr>
<tr>
<td>10</td>
<td>0.02</td>
<td>21</td>
<td>19</td>
<td>2</td>
<td>1.003344</td>
<td>1.034413</td>
</tr>
<tr>
<td>10</td>
<td>0.03</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>1.002389</td>
<td>1.018578</td>
</tr>
<tr>
<td>10</td>
<td>0.04</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1.002415</td>
<td>1.025875</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1.000944</td>
<td>1.008980</td>
</tr>
<tr>
<td>10</td>
<td>0.08</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.000325</td>
<td>1.003012</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000172</td>
<td>1.001859</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000152</td>
<td>1.001391</td>
</tr>
</tbody>
</table>

Figure 8 shows the total number of failures for different densities of traffic as a function of the speed range $s_{\text{lon}} \in [0., 1.]$. For the low and medium densities (10, 20, 30 or 50 aircraft) the algorithm is very efficient when the speed can be modified by at least 10% but suddenly becomes totally ineffective when the speed is constrained in a smaller interval. This is also true in the high density ranges: for 100 aircraft the solver becomes efficient for speed changes greater than 15%, which is still low. Even for the densest cases (120 and 130 aircraft), a speed change of 25% is enough to solve the majority of cases.

These figures give a better understanding of the efficiency of such algorithms in robotics where agents can change speeds very easily. Autonomous approaches also works very well to model car or pedestrian traffic for the same reason. These
results show as well how such algorithms lose their efficiency when the speed is constrained. In particular, the European ERASMUS project [24], [25], which aims at automatically solving conflict by slight speed changes, set that only half of the scenarios are solved with the lowest density in this range. We can also remark that with no speed change, that only the lowest density allows a solution for every conflict acceptable for aircraft, which is about \( \pm 5\% \) (whereas it was not the case for the ERASMUS project).

When we look at the reasons for failure, in very low densities without speed change \( (s_{\text{lon}} \leq 0.1) \), most of them are border violations, which means that the autonomous approach fails to find a solution that could keep every aircraft inside the designed sector. This is even the case with only 10 aircraft for half of the runs when no speed change is allowed.

Figure 8 and 11 show the number of failures due to border violations and those due to remaining conflicts. For high densities \( (\geq 100 \text{ aircraft}) \), the failures are mainly due to remaining conflicts. For low densities \( (\leq 30 \text{ aircraft}) \) the failures are mainly due to border violations, because there is enough space to keep aircraft separated, but they tend to “push” each other out of the sector.

Figure 9 shows the trajectory lengthening in the different cases. It increases with the density, which is expected, and decreases when the speed constraint loosens because aircraft can accelerate and compensate for the delays caused by conflicts.

### B. Influence of the turning angle constraint \( s_{\text{lat}} \)

When \( s_{\text{lon}} = 0 \), \( s_{\text{lat}} \) can be seen as the tangent of the turning angle. In this section, we set \( s_{\text{lon}} = 0.01 \) (the speed change cannot exceed 1\%) and assess the behavior of the autonomous algorithm for different densities and values of \( s_{\text{lat}} \). Table II

<table>
<thead>
<tr>
<th>nb acft</th>
<th>8lon</th>
<th>Tot Fail</th>
<th>Border Viols</th>
<th>Rem Conf</th>
<th>mean delay</th>
<th>max delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.05</td>
<td>100</td>
<td>8</td>
<td>92</td>
<td>1.024383</td>
<td>1.032395</td>
</tr>
<tr>
<td>100</td>
<td>0.08</td>
<td>100</td>
<td>16</td>
<td>68</td>
<td>1.023625</td>
<td>1.031094</td>
</tr>
<tr>
<td>100</td>
<td>0.1</td>
<td>100</td>
<td>24</td>
<td>66</td>
<td>1.021868</td>
<td>1.027560</td>
</tr>
<tr>
<td>100</td>
<td>0.12</td>
<td>100</td>
<td>66</td>
<td>53</td>
<td>1.015557</td>
<td>1.024916</td>
</tr>
<tr>
<td>100</td>
<td>0.2</td>
<td>100</td>
<td>30</td>
<td>79</td>
<td>1.011124</td>
<td>1.021002</td>
</tr>
<tr>
<td>100</td>
<td>0.3</td>
<td>100</td>
<td>19</td>
<td>91</td>
<td>1.011288</td>
<td>1.021665</td>
</tr>
<tr>
<td>100</td>
<td>0.4</td>
<td>100</td>
<td>14</td>
<td>103</td>
<td>1.010339</td>
<td>1.022460</td>
</tr>
<tr>
<td>100</td>
<td>0.5</td>
<td>100</td>
<td>11</td>
<td>103</td>
<td>1.010307</td>
<td>1.017107</td>
</tr>
<tr>
<td>100</td>
<td>0.6</td>
<td>100</td>
<td>6</td>
<td>106</td>
<td>1.009770</td>
<td>1.014701</td>
</tr>
<tr>
<td>120</td>
<td>0.05</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>1.024330</td>
<td>1.027306</td>
</tr>
<tr>
<td>120</td>
<td>0.08</td>
<td>100</td>
<td>5</td>
<td>95</td>
<td>1.024330</td>
<td>1.027306</td>
</tr>
<tr>
<td>120</td>
<td>0.1</td>
<td>100</td>
<td>8</td>
<td>92</td>
<td>1.024330</td>
<td>1.027306</td>
</tr>
<tr>
<td>120</td>
<td>0.12</td>
<td>100</td>
<td>16</td>
<td>68</td>
<td>1.024330</td>
<td>1.027306</td>
</tr>
<tr>
<td>120</td>
<td>0.15</td>
<td>100</td>
<td>24</td>
<td>66</td>
<td>1.024330</td>
<td>1.027306</td>
</tr>
<tr>
<td>120</td>
<td>0.2</td>
<td>100</td>
<td>66</td>
<td>53</td>
<td>1.015557</td>
<td>1.024916</td>
</tr>
<tr>
<td>120</td>
<td>0.25</td>
<td>100</td>
<td>30</td>
<td>79</td>
<td>1.011124</td>
<td>1.021002</td>
</tr>
<tr>
<td>120</td>
<td>0.3</td>
<td>100</td>
<td>19</td>
<td>91</td>
<td>1.011288</td>
<td>1.021665</td>
</tr>
<tr>
<td>120</td>
<td>0.4</td>
<td>100</td>
<td>14</td>
<td>103</td>
<td>1.010339</td>
<td>1.022460</td>
</tr>
<tr>
<td>120</td>
<td>0.5</td>
<td>100</td>
<td>11</td>
<td>103</td>
<td>1.010307</td>
<td>1.017107</td>
</tr>
</tbody>
</table>

| 130    | 0.05 | 100      | 0            | 100      | 1.024330   | 1.027306 |
| 130    | 0.08 | 100      | 0            | 100      | 1.024330   | 1.027306 |
| 130    | 0.1  | 100      | 6            | 94       | 1.024330   | 1.027306 |
| 130    | 0.12 | 100      | 5            | 94       | 1.024330   | 1.027306 |
| 130    | 0.15 | 100      | 14           | 80       | 1.024330   | 1.027306 |
| 130    | 0.2  | 100      | 75           | 57       | 1.024330   | 1.027306 |
| 130    | 0.25 | 100      | 3            | 46       | 1.024330   | 1.027306 |
| 130    | 0.3  | 100      | 23           | 23       | 1.024330   | 1.027306 |
| 130    | 0.35 | 100      | 15           | 15       | 1.024330   | 1.027306 |
| 130    | 0.4  | 100      | 13           | 13       | 1.024330   | 1.027306 |
| 130    | 0.5  | 100      | 11           | 11       | 1.024330   | 1.027306 |
| 130    | 0.6  | 100      | 9            | 9        | 1.024330   | 1.027306 |

<table>
<thead>
<tr>
<th>10 aircraft</th>
<th>20 aircraft</th>
<th>30 aircraft</th>
<th>50 aircraft</th>
<th>100 aircraft</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.04</td>
</tr>
</tbody>
</table>

**TABLE II**

**TOTAL NUMBER OF FAILURES DUE TO BORDER VIOLATIONS OR REMAINING CONFLICTS, MEAN AND MAX DELAYS FOR 100, 120 OR 130 AIRCRAFT AND VARIOUS SPEED RANGE \( s_{\text{lon}} \) VALUES.**
Fig. 10. Failures due to border violations (as a function of $s_{lon} \in [0, 1]$).

Fig. 11. Failures due to remaining conflicts (as a function of $s_{lon} \in [0, 1]$).

Fig. 12. Trajectory lengthening (as a function of $s_{lon} \in [0, 1]$).

shows the experimental results for densities varying from 10 to 100 aircraft and various $s_{lat}$ values.

Figure 13 shows the number of total failures for different densities as a function of $s_{lat}$. For high densities, no solution is found, whatever the turning angle. For medium density (50 aircraft) the number of failures slowly decreases when $s_{lat}$ increases but it stays high ($\geq 70$). For low densities, the number of failures also decreases but never reaches zero.
Evolutionary Algorithm, in order to show the necessity to take into account the modeling in order to allow the same continuous maneuvers when the speed is constrained. The comparison algorithm derived from robotics and able to handle thousands of agents when speeds and turning angles of the robots are not constrained. Our experiments showed how the speed constraint severely degrades the algorithm’s performance, even when the density of traffic is low. Even in low densities, some scenarios failed to find solutions when the speed was constrained: aircraft converging with a small angle tend to “push” each other out of the sector. In contrast, the turning angle constraint has less of an impact on the resolution.

Finally, figure 16 shows the mean trajectory lengthening as a function of $s_{lat}$. As expected, the lengthening increases both with the traffic density and $s_{lat}$.

### III. Comparison with a Centralized Approach on an Example

In this section, we use an example to compare the autonomous algorithm to a centralized approach based on an Evolutionary Algorithm, in order to show the necessity to coordinate maneuvers when the speed is constrained. The algorithm used is the same as described in [13]. We changed the modeling in order to allow the same continuous maneuvers as for the autonomous approach. At each time step, an evolutionary algorithm is used to optimize the speeds that guarantee no conflict for $\tau$ seconds, and check the effectiveness of the resolution (verifying that aircraft that need to pass each other really do).

The example shows 15 aircraft converging on the center of a circle (with a 5 degree angle of convergence between aircraft). Figure 17 shows the solution found with the autonomous algorithm and no speed change allowed. Figure 18 shows the solution found with the autonomous algorithm and 20% speed change allowed. Figure 19 shows the solution found with the centralized approach and no speed change allowed. Here we can observe the benefit of using centralized information to coordinate maneuvers and help aircraft pass each other without slowing or accelerating them.

### Conclusion and Future Work

In this article, we presented an autonomous conflict resolution algorithm derived from robotics and able to handle thousands of agents when speeds and turning angles of the robots are not constrained. Our experiments showed how the speed constraint severely degrades the algorithm’s performance, even when the density of traffic is low. Even in low densities, some scenarios failed to find solutions when the speed was constrained: aircraft converging with a small angle tend to “push” each other out of the sector. In contrast, the turning angle constraint has less of an impact on the resolution.
We designed a particularly complex example to show the consequence of a speed constraint on aircraft converging with a small angle. When the speed constraint is relaxed, the algorithm finds a solution, but it fails when no speed change is allowed. We have also shown that a centralized approach that coordinates maneuvers can handle the complex case. These findings strongly suggest that Air Traffic Management needs centralized coordination because of the constrained speed environment.

Building reliable resolution tools for solving conflicts is very challenging. Van den Berg et al.'s algorithm is very efficient in a non-constrained context. For now, it modifies speed vectors of both aircraft by “sharing efforts” to reach the no-conflict area. This rule is simple and does not require any coordination between aircraft. With a coordination process, it could be modified to increase chances of solving more complex conflicts. This needs to be investigated.

The centralized approach used in section III needs to be tested on the scenarios used for measuring the autonomous approach, and not only on the complex case presented. This would help show the advantage of keeping centralized control of air traffic more generally. Results could be used as a reference for comparing with further improvements of this autonomous approach or with other autonomous approaches.

References


Nicolas Durand graduated from the École Polytechnique de Paris in 1990 and the École Nationale de l’Aviation Civile (ENAC) in 1992. He has been a design engineer at the Centre d’Études de la Navigation Aérienne (now DSNA/DTI R&D) since 1992, holds a Ph.D. in Computer Science (1996) and got his HDR (french equivalent of tenure) in 2004. He is currently professor at the ENAC/MAIAA (Applied Maths, Computer Science, Automatic applied to Aviation) laboratory.

Nicolas Barnier is a lecturer and research assistant at ENAC. He graduated from ENAC as an engineer in 1997, and received a Ph.D. in computer science in 2002 from the University of Toulouse. He is one of the authors of FaCiLe, an open source Constraint Programming library for the functional language OCaml. His research interests focus on Constraint Programming, Local Search and Combinatorial Optimization in general.