A New Framework for Solving En Route Conflicts

Cyril Allignol, Nicolas Barnier, Nicolas Durand, Jean-Marc Alliot

A New Framework for Solving En route Conflicts

Cyril Allignol, Nicolas Barnier, Nicolas Durand, and Jean-Marc Alliot

The en route conflict resolution problem has been modeled in many different ways, generally depending on the tools proposed to solve it. For instance, with purely analytic mathematical solvers, models tend to be very restrictive to respect the inherent limitations of the technology.

This paper introduces a new framework that separates the model from the solver so as to be able to: first, enhance the model with as many refinements as necessary to comply with operational constraints; second, compare different resolution methods on the same data, which is a crucial aspect of scientific research.

To this aim, our framework generates a benchmark of conflict resolution problems built with various scenarios involving different numbers of aircraft, levels of uncertainties and numbers of maneuvers. We then compare two different optimization paradigms, Evolutionary Algorithm and Constraint Programming, which can efficiently solve difficult instances in near real time, to illustrate the usefulness of our approach.

INTRODUCTION

An effective conflict solver relies on a realistic trajectory prediction. Today, because of different uncertainty sources such as wind and aircraft mass, air traffic management systems are unable to predict the future positions of aircraft with a good accuracy and must take

Cyril Allignol, Nicolas Barnier, and Jean-Marc Alliot are with the Institut de Recheche en Informatique de Toulouse; Nicolas Durand is with the Ecole Nationale de l'Aviation Civile (ENAC)

Received: June 19, 2013; accepted September 25, 2013.

Air Traffic Control Quarterly, Vol. 21(3) pp. 1–21 (2013)
 CCC 1064-3818/95/030163-20

into account all these uncertainties to choose the best trajectories in terms, first, of safety and then, efficiency. These certainties probably explain why the short-term traffic resolution system still relies on human expertise and is not yet automated.

Much research has been done on conflict detection and resolution and many papers present models that are so impractical that they strengthen the readers' beliefs that automating the conflict detection and resolution task is unrealistic in the near-term. For example, the approach using repulsive forces described in [Zeghal, 1993] or the B-spline approximation model of [Delahaye et al., 2010] are very interesting on a mathematical level but could hardly be implemented in an operational context. They suppose continuous heading changes, which Flight Management Systems (FMS) are unable to exploit, and do not take uncertainties into account. Pallottino's approach [Pallottino et al., 2002] using mixed integer linear programming (as [Vela et al., 2009, Alonso-Ayuso et al., 2011, Rey et al., 2012]) relies on constant speed trajectories that are changed all at once. None of these approaches could deal with realistic trajectory models able to handle evolutive aircraft or trajectory uncertainties.

Other approaches like [Durand et al., 1996, Granger et al., 2001] propose to solve conflicts using Evolutionary Algorithms, relying on more realistic models built upon the Base of Aircraft Data (BADA) developed and maintained by EUROCONTROL. These models introduce uncertainties on aircraft speed, climb and descent rate, thus the solver needs to compute many alternative trajectories in real time. Nevertheless, the solver is quite efficient as it can handle complete days of traffic in the European airspace. These algorithms, however, are difficult to compare with other methods because the conflict detection is embedded in the solver. This problem also occurs in Erzberger's approach [Erzberger, 1997], where most of the expertise is focused on the trajectory and maneuver model. Once more, the presented results can hardly be compared with other algorithms because the resolution maneuver generator is embedded in the solver.

We propose a new framework to deal with conflict resolution, which, from a given scenario, trajectory model, and maneuver model, computes a 4D-matrix indexed by aircraft pairs and maneuvers (i.e., trajectories) pairs that provides all necessary data to solve the problem. Hence, the detection and maneuver model is separated from the resolution, which enables us to compare the behavior of various algorithms at the same time. As the conflict resolution problem is highly combinable, and as large instances can, therefore, be very difficult to optimize [Durand and Granger, 2003], it is of utmost importance to be able to assess the relative merits of solvers, even if finding the optimal solution is often not required in a real-time context (a "good" conflict-free solution can be sufficient). This clean separation between the underlying models (trajectory and maneuver) and resolution

allows us to publish a benchmark, which can be retrieved at the following address:

`clusters.recherche.enac.fr`

for the scientific community to compare the results of various resolution algorithms. In our benchmark framework, finding the future positions can be done using any simulator and can take into account different uncertainty sources such as wind, heading change, and beginning and ending maneuver positions.

Once the future positions for all possible maneuvers are found, a simple algorithm can detect conflicts for each pair of aircraft and store this information in a 4D conflict matrix C, the first two indexes specifying the pair of aircraft involved and the next two the concerned maneuvers. For example, $C_{i,j,k,l}$ returns *true* if maneuver k of aircraft i and maneuver l of aircraft j are conflicting, and *false* otherwise. When solving conflicts, instead of recomputing the trajectory positions of each aircraft, solvers can refer to this 4D-matrix quickly, in constant time complexity.

We then illustrate how resolution times and costs of different solvers can be fairly compared within our benchmark framework. Two resolution algorithms using different optimization paradigms have been implemented: a metaheuristic, namely an Evolutionary Algorithm [Michalewicz, 1992], which is able to handle over-constrained or very large instances, but cannot provide optimality proofs, and a Constraint Program [Van Hentenryck, 1995], which has the converse properties (may be stalled while backtracking for large scale problems, but able to provide proofs of optimality or absence of solution for reasonable ones).

The next section of this paper introduces the model that was chosen to build the trajectories of our benchmark framework. We particularly detail its uncertainty model and how the convex hulls of trajectories are built. The following section describes the method used to build conflicting scenarios with different sizes (number of aircraft) and levels of uncertainties, before presenting the detection algorithm that builds the 4D conflict matrix. We then detail two approaches for the resolution of conflicts, an Evolutionary Algorithm and a Constraint Program, to illustrate their comparison with our benchmark framework, where some experimental results are analyzed.

TRAJECTORY PREDICTION MODEL

In this section we give an example of a trajectory prediction tool that can be used to build the aircraft positions at each time step according to the chosen maneuver options and the uncertainties taken into account. To constrain the search space to a “reasonable” size, only a limited number of maneuvers, compatible with current ATC practice

and FMS capabilities, is defined for each aircraft involved in a conflict. Then, each pair of maneuvers for two different aircraft is tested to check for confliction.

Moreover, our model can handle various degrees of uncertainties by considering the future positions of aircraft not simply as mere 2D-points in the airspace but as growing convex envelopes representing all its possible positions. Loss of separation between aircraft is then detected by computing the minimal distance between their two envelopes.

Maneuvers

In our trajectory prediction model, a discretization of time into steps of duration τ is used to describe maneuvers. τ is chosen small enough to detect every significant conflict in the application. Only conflicts at the fringe of the detection volume and lasting less than τ seconds could be missed, as indicated by the formula (detailed in [Barnier and Allignol, 2012]:

$$D > N_h \sin\left(\arccos\left(\frac{\tau V_{\max}}{N_h}\right)\right)$$

were N_h is the horizontal separation norm, $V_{\max}$ is the top speed of the aircraft fleet and D is the closest approach distance during the conflict.

In our experiments, we chose $\tau = 3$ s (given $N_h = 5$ nmi and $V_{\max} = 600$ kn), so that conflicts might only occur for 5 nmi $> D > 4.97$, which is commonly accepted for fast-time simulations as such routing schemes are not allowed by standard ATM practice.

Trajectories are defined in the horizontal plane, but the scenarios could be extended easily to the vertical dimension if we used a proper flight simulator. Initial routes are defined by a list of points. The first point O is the origin and the last point D is the destination (e.g. a segment of trajectory between two waypoints). Aircraft fly from point to point and are able to correct the lateral error to the original trajectory thanks to their FMS. This means that in the further examples, the associated uncertainty does not increase with time.

Various other sources of uncertainties cannot be reduced by current FMS features and must be taken into account in our model. Hence, aircraft speeds are subject to a ε_s error such that future positions of aircraft are spread over a range that grows with time.

In our trajectory model, maneuvers (i.e., heading changes) are engaged on a point of the initial trajectory referenced by the decision variable d_0, which represents the curvilinear distance from the origin O. Because of uncertainties on the exact location of the turn, a distance error ε_0 is added around this point. This means that the aircraft may start the maneuver ε_0 nautical miles before or after d_0.

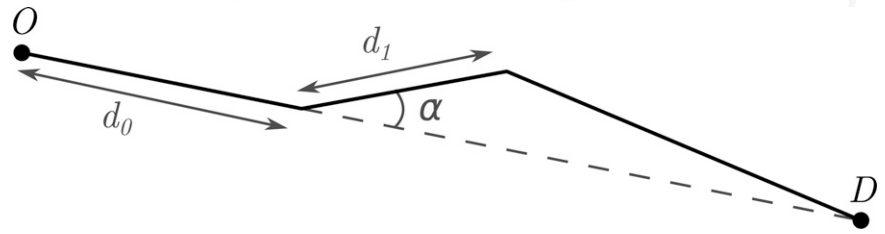

Figure 1. Maneuver model.

An uncertainty ε_α is also associated to the heading change angle α at the turning point corresponding to d_0. Then the maneuver ends at a curvilinear distance d_1 from d_0 (i.e. at $d_0 + d_1$ from the origin O) with an associated error ε_1, when the aircraft returns towards its destination point D.

These simple maneuvers, depicted in figure 1, are representative of current Air Traffic Control practice and can be implemented easily by pilots and current FMS technologies (cf. [Granger et al., 2001]), unlike continuous maneuvers at arbitrary angles and distances that are used in many conflict resolution models [Zeghal, 1993; Delahaye et al., 2010].

To limit the number of maneuvers created, and thus the size of the search space, d_0 can only take a limited number n_0 of values (typically $n_0 = 5$ in the experimental benchmark presented in section 5). The heading change α can also take $n_\alpha = 7$ different values in our benchmark, i.e. 0, 10, 20 or 30 degrees to the left or the right of the current heading, and the number of values for the distance of the returning point d_1 is also limited by n_1 (typically $n_1 = 5$).

If we consider 5 values for d_0, 5 values for d_1 and the 6 possible angles (there is no need to combine a null heading change $\alpha = 0$ with various d_0 and d_1 values, so that only one maneuver is added when the aircraft is not deviated), the number of maneuvers per aircraft is:

$$n_{\text{man}} = n_0 \times n_1 \times (n_\alpha - 1) + 1$$

So for the benchmark presented in the later section on Benchmark Generation Scenarios: $n_{\text{man}} = 5 \times 5 \times 6 + 1 = 151$.

For an instance with n aircraft, the search space is then of size n_{man}^n, i.e. $\approx 6.10^{21}$ for a 10-aircraft instance (almost 4.10^{43} for 20 aircraft).

Decision Variables

To simplify the access to the conflict matrix C and reduce the number of combinations to the useful ones (e.g., only one possible maneuver for $\alpha = 0$), the three decision variables d_0, α and d_1 associated with aircraft i are aggregated into a single decision variable m_i by a

bijection from the allowed triples to interval [1, n_{man}]. We call M the set of decision variables of the problem:

$$M = \{m_i, i \in [1, n]\} \tag{1}$$

Cost

The maneuver cost of our model is straightforwardly computed from the decision variables. Values of d_0 are enumerated by an index k_0 varying in [1, n_0], values of d_1 by index k_1 in [1, n_1] and angles α of value 10, 20 or 30 degrees right or left, are respectively indexed by k_α in $[1, \frac{n_\alpha}{2}]] >]$. For our benchmark problems, the cost of a maneuver m_i for aircraft i is then defined as follows:

$$\text{cost}_{\text{man}}(m_i) = \begin{cases} 0 & \text{if } \alpha = 0 \\ (n_0 - k_0)^2 + k_1^2 + k_\alpha^2 & \text{otherwise} \end{cases} \tag{2}$$

where k_0, k_1 and k_α are the indexes corresponding to maneuver m_i. This cost is null whenever an aircraft is not maneuvered.

Furthermore, this cost function ensures the following properties:

1. any maneuver is more costly than no maneuver;
2. maneuvers should start as late as possible;
3. maneuvers should be as short as possible;
4. the angle should be as small as possible.

In a real environment, the cost function should be adapted to the aircraft performance model or to other criteria, including controllers' preferences and fuel consumption. This paper, however, aims at giving a framework that dissociates the solver method from the problem itself, so as to provide the scientific community (which may be unfamiliar with ATM and conflict resolution) with the simplest possible framework in which to compare different solvers on our benchmark.

Given an instance with n aircraft, we define the cost of a solution as the sum of the maneuvers costs:

$$\text{cost} = \sum_{i=1}^{n} \text{cost}_{\text{man}}(m_i) \tag{3}$$

Handling Uncertainties

We shall now describe how the trajectories envelopes are built to be able to detect conflicts between two maneuvers for two different aircraft, while taking various uncertainties into account.

In our framework, the maneuvers description is stored in a table that defines the possible future positions of the aircraft for each aircraft and each maneuver at every time step. These positions are represented by their convex hull (i.e., the smallest convex set

containing all possible positions at a given time step), which is computed with Graham's algorithm [Graham, 1972].

Each aircraft position is described at multiples of the time step τ (i.e., 0, τ, 2 τ, 3 τ . . .) by three convex hulls corresponding to the three possible states of the aircraft:

- S_0 if it has not been maneuvered yet;
- S_1 if it is currently maneuvered;
- S_2 if it is heading towards its destination after a maneuver.

Once the three convex hulls are defined for every time step, if there are any instances of multiple states occurring within a single step (e.g., around trajectory turning points), they are merged into a single encompassing convex hull for that step.

We first start with one point representing the current position of the aircraft at $t = 0$. To build the possible positions at $t + \tau$, we take into account every extreme position of the three convex hulls at time t and calculate the future possible positions of each point. During this process, some points stay in the same state while others change near the turning points of the trajectory. Moreover, some points may generate two different future positions in two different states. For example, a point in state S_0 (before any maneuver) may reach $d_0 - \varepsilon_0$ at the next step if the aircraft flies at the fastest possible speed according to the amount of uncertainty taken into account by parameter ε_0. It will then change heading and be in state S_1. The same point may as well fly at the lowest possible speed and stay in state S_0. After each movement, the convex hull of the cloud of points created is computed for each state. At the very end of the process, the convex hull of the whole trajectory is calculated for each time step.

Figure 2 displays a maneuver with the different states involved F2
(each state being associated to a unique plot style). At each time step, the thin gray line shows the convex hull encompassing all possible positions. The conflicts will then be detected among such envelopes by computing their minimal distance.

Importantly, notice that any traffic simulator using any kind of uncertainty hypothesis could be used to build the trajectory prediction

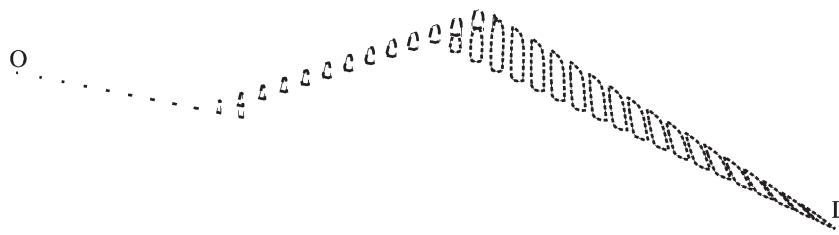

Figure 2. An example of trajectory prediction. Different plot styles correspond to states S_0, S_1 and S_2; gray parts represent the convex hulls.

for the aircraft and for the maneuver options. Different aircraft could have different uncertainties and different maneuver options according to their ability to follow a route. We simply need a convex hull of the possible future positions of an aircraft at every time step of the trajectory.

This approach can easily be generalized to the third dimension (vertical plane), taking into account uncertainties on the climbing rate of the aircraft. Convex 3D- volumes would thus be defined and conflicts detected according to the distance between them.

BENCHMARK GENERATION

The trajectory prediction presented in the previous section is used to produce the data for the proposed benchmark. To generate an instance, two consecutive processes are involved: first the creation of conflicting scenarios, then the detection of conflicts.

Scenarios

In the experimental results presented in the Results section, instances of four different sizes have been considered, involving $n =$ 5, 10, 15 and 20 aircraft, with three levels of uncertainties. For each combination, 10 scenarios of aircraft converging to the center of the considered airspace volume were randomly built, which is the most penalizing situation for conflict resolution, as stated in [Durand, 2004]. For each scenario, speeds are chosen from 384 kn to 576 kn (i.e., 20% variation around a typical speed of 480 kn). The aircraft initial positions are chosen on a 70 nmi radius circle and are noised within a 20 nmi-side square. The initial heading is also noised with a value chosen in $[-1, 1]$ radians ($\approx \pm 60^\circ$). Figure 3 illustrates this geometry on an instance with four aircraft.

A total of 40 scenarios were built to compare the algorithms. For each scenario, three levels of uncertainty are defined. The lower level of uncertainty ε_{low} takes into account $\varepsilon_s = 1\%$ of error on the aircraft speed, $\varepsilon_0 = 1$ nmi of error on the location of the turning point, $\varepsilon_\alpha = 1^\circ$ on the angle of the turn and $\varepsilon_1 = 1$ nmi of error on the location of the returning point. The medium level of uncertainty ε_{med} doubles every value: $\varepsilon_s = 2\%$, $\varepsilon_0 = 2$ nmi, $\varepsilon_\alpha = 2^\circ$ and $\varepsilon_1 = 2$ nmi. Finally, the higher level of uncertainty $\varepsilon_{\text{high}}$ triples the lower uncertainty values: $\varepsilon_s = 3\%$, $\varepsilon_0 = 3$ nmi, $\varepsilon_\alpha = 3^\circ$ and $\varepsilon_1 = 3$ nmi. 120 scenarios are thus built as a proposed benchmark basis and we next detail how conflict are detected to complete the framework description.

Conflict Detection

Once the trajectory predictions computed and stored, the 4D conflict matrix C can be built. To simplify the access to the matrix and reduce

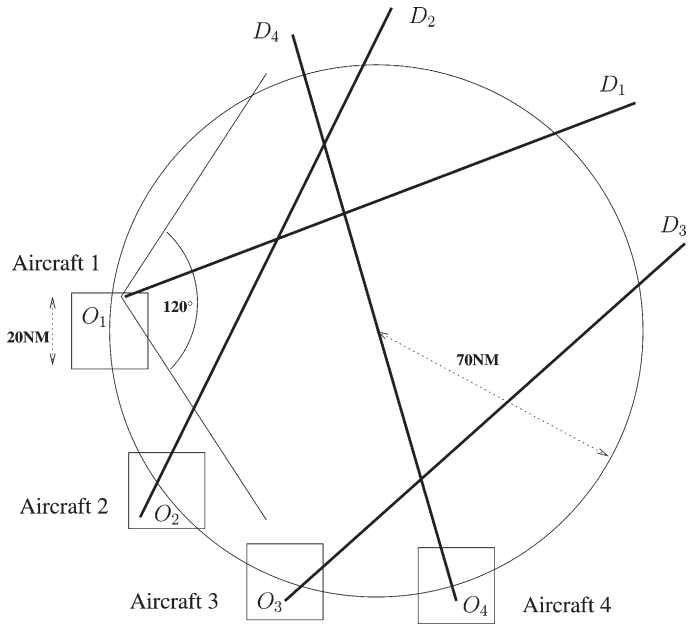

Figure 3. Geometry of the conflict scenario generation (with four aircraft).

the number of combinations to the useful ones (e.g. only one possible maneuver for $\alpha = 0$), the three decision variables d_0, α and d_1 are aggregated in a single decision variable by a bijection from the allowed triples to interval $[1, n_{\text{man}}]$. Then, for each pair of aircraft (i,j) and each pair of maneuver options (k,l) (where k is a maneuver option for aircraft i and l for aircraft j), we test if maneuvers k and l generate a conflict. In this case, $C_{i,j,k,l} = 1$ (i.e. *true*), otherwise $C_{i,j,k,l} = 0$ (i.e. *false*). Furthermore, the matrix is symmetric along its two first dimensions, since a conflict between i and j is equivalent to a conflict between j and i, so we only consider pairs of aircraft such that $i < j$.

To detect a conflict, the distance between the two envelopes representing the possible positions of aircraft i and j is computed and compared to the standard separation norm (5 nmi). For every time step, the algorithm is divided in three stages:

1. Check if a vertex of convex hull k is inside convex hull l, or if a vertex of convex hull l is inside convex hull k.
2. Otherwise, check if two edges of convex hulls k and l intersect.
3. Otherwise, check the distance between every vertex of convex hull k and every edge of convex hull l, or every vertex of convex hull l with every edge of convex hull k. As soon as one of the distances is smaller than the separation standard, $C_{i,j,k,l}$ is set to 1.

This calculation is the most time consuming of the problem generation because the number of pairs tested is big. For example, a 20-aircraft conflict with 151 maneuvers per aircraft generates

$\frac{20\times19}{2} = 190$ pairs of aircraft for which $151^2 = 22,801$ pair of maneuvers must be tested. A total of $4,332,190$ pairs of maneuvers must be tested to build the conflict matrix.

This operation, however, can be parallelized very easily. For instance, different processors can be used to test different pairs of maneuvers. Thus, the computation time can be reduced drastically. We give different examples in section 5 of the time required to compute $C_{i,j,k,l}$ as a function of the number of processors. Furthermore, sweep-line techniques [de Berg et al., 1998] could be used to lower the time complexity of the edge intersection checks performed during the second step of our convex hull distance algorithm. Eventually, a preliminary filtering by approximating the envelopes with simple enclosing boxes can spare many convex hull intersection checks.

Now that our framework is equipped with all the necessary precomputed data needed to implement a conflict solver independently of the trajectory generation or the conflict detection, we describe two different approaches to solve the conflict scenarios of the proposed benchmark in the next section.

CONFLICT RESOLUTION

In this section, we propose two methods to resolve the conflict scenarios generated with our benchmark framework. The first one, an Evolutionary Algorithm, is a metaheuristic that mimics natural evolution to explore the search space. The second one, Constraint Programming, is based on an efficient systematic search of the solution space, which proves the optimality (or the absence) of a solution.

Evolutionary Algorithm

Principles. Our Evolutionary Algorithm (EA), described in algorithm 1, follows classical Evolutionary Computation principles [Goldberg, 1989, Michalewicz, 1992].

First, a population of points in the state space is randomly generated. Then, we compute for each population element the value of the

Algorithm 1 Evolutionary algorithm (EA)

1: Initialize population
2: **while** termination criterion is not met **do**
3: Evaluate *raw fitness* of population elements
4: Apply *scaling* and *sharing* operations on *raw fitness*
5: Select new population w.r.t. *new fitness* criterion
6: Replace some elements by *mutation* and *crossover*
7: **end while**
8: Return best elements of population

function to optimize, which is called *fitness*. In a second step, we select[1] the best individuals in the population according to their fitness. Afterwards, we randomly apply classical evolutionary operators, i.e. *crossover* and *mutation*, to diversify the population (they are applied with respective probabilities P_c and P_m). At this step a new population has been created and we apply the process again in an iterative way.

Sharing Improvement. Our problem is very combinatorial and may have many different optimal solutions.[2] To find most of these solutions and to avoid local optima, the *sharing* process introduced by Yin and Germay [Yin and Germay, 1993] is used. This improvement can be computed efficiently in $\Theta(p \log p)$ time complexity (instead of $\Theta(p^2)$ for the classical sharing process), where p is the size of the population.

A sharing process requires the definition of a distance between two chromosomes (two trajectory sets) to group alike population elements in the same *cluster*, according to a threshold parameter controlling the size and number of clusters. For the sake of simplicity, the distance implemented in our EA returns only two values: *true* if the elements (set of trajectories) are identical and *false* if otherwise. The fitness of elements belonging to the same cluster is then divided by the size of the cluster to avoid an over-representation of a particular solution in the population and encourage diversification.

Fitness Function. The fitness function of our EA is very basic and does not aim at taking into account fuel consumption or controllers' preferences. We just focus on finding a conflict- free set of heading changes starting as late as possible, with the smallest deviation length and heading change.

The fitness function is then defined by two cases, depending on the presence (first case) or absence (second case) of remaining conflicts in the solution:

$$F = \begin{cases} \dfrac{1}{2 + \displaystyle\sum_{i<j} C_{i,j,m_i,m_j}} & \text{if } \exists (i,j), i<j, C_{i,j,m_i,m_j} \neq 0 \\ \dfrac{1}{2} + \dfrac{1}{1+\text{cost}} & \text{if } \forall (i,j), i<j, C_{i,j,m_i,m_j} = 0 \end{cases}$$

where cost, defined by equation 3 in section 2.3, represents the cost of a solution.

[1] Selection aims at reproducing better individuals according to their fitness. We tried two kinds of selection process, "Roulette Wheel Selection" and "Stochastic Remainder Without Replacement Selection" (described in [Goldberg, 1989] for example); the latter always works out better.

[2] Finding several solutions is very interesting because the controller may choose among several options or negotiate them with pilots, keeping controllers and pilots alike in the decision making process.

Moreover, this fitness function guarantees that if a chromosome value is larger than $\frac{1}{2}$, no conflict occurs, such that the cost of proper solutions is strictly greater than the cost of conflicting ones. If a conflict remains, the fitness does not take into account the cost of the maneuvers, allowing the EA to focus the search for conflict-free solutions first, regardless of the quality of the maneuvers involved.

Adapted Crossover and Mutation. EAs are very versatile because they do not require much information on the objective function. Non-specific classical operators, however, used by Gruber, Alliot, and Schoenauer in [Alliot et al., 1993] did not produce satisfactory results on our benchmark.

Nonetheless, in the case of the conflict resolution problem, we do know many properties about the fitness function, and they can be useful in creating adapted crossover and mutation operators. Durand, Alliot and Noailles describe such operators in [Durand et al., 1994].

The crossover operator, tailored to benefit from the structure of functions defined as a sum of positive terms, is described on figure 4. F4 After choosing two parents A and B, we compare the number of conflicts remaining for both sets of maneuvers and choose the maneuver from the parent having the smallest number of conflicts. When both maneuvers generate the same number of conflicts, we pick up randomly the maneuver from parent A and parent B.

The mutation operator is described on figure 5. After choosing a F5 chromosome, an aircraft is mutated (on figure 5, aircraft 4 is chosen).

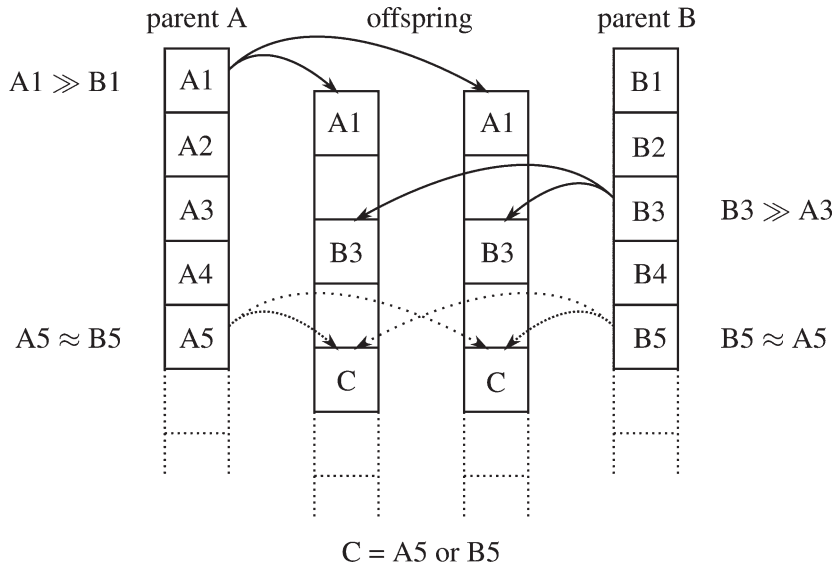

Figure 4. Adapted crossover operator.

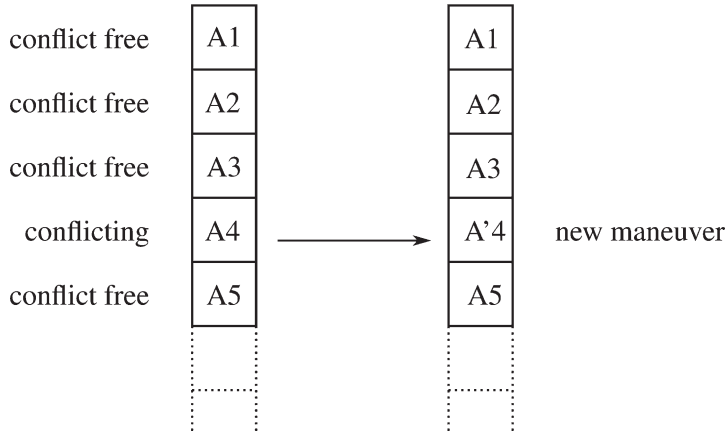

Figure 5. Adapted mutation operator.

Maneuvers generating conflicts for the parent are chosen in priority and changed to favor conflict-free maneuvers in the offspring.

These operators are more deterministic at the beginning of the optimization, when many conflicts remain in the population, so that a solution without conflict can be found quickly. When conflict-free solutions become sufficiently numerous, more randomness is allowed, and other parts of the search space can be explored.

Constraint Programming

Constraint Programming (CP) is a versatile optimization technology based on the Constraint Satisfaction Problem (CSP) formalism that emphasizes the satisfaction of combinatorial *constraints* (i.e., arbitrary relations over a set of decision variables). CP offers a clean separation between the modeling language and the resolution algorithms, enabling to develop solvers quickly and incrementally and to experiment with various search strategies without changing the model. See [Van Hentenryck, 1995] for example, where more details on the CP technology can be found.

CSP Model. The set M of decision variables of the CSP is the one defined by equation 1, where each variable m_i is the index of the maneuver for aircraft i and, thus, takes a value in $[1, n_{\text{man}}]$.

The constraints are expressed as *binary constraints*, i.e., constraints involving exactly two variables. For a given couple of aircraft i and j ($i < j$), the constraint c_{ij} between variables m_i and m_j is defined as the set:

$$c_{ij} = \left\{ \left(m_i^k, m_j^l \right) \quad \text{s.t.} \quad C_{i,j,k,l} = 1 \right\} \tag{4}$$

where m_i^k and m_j^l are respectively the k-th and the l-th value of interval $[1, n_{\text{man}}]$ of the maneuvers available for aircraft i and j. c_{ij}

therefore describes all couples of maneuvers that *cannot* be performed by aircraft i and j without resulting in a conflict.

We denote by $|c_{ij}|$ the cardinal of the constraint c_{ij}, i.e. the number of forbidden couples of maneuvers.

Solution Search. The exploration of the search space is based on an enhanced version of a systematic tree-search algorithm called *backtracking*, where an inference phase prunes the unfeasible values of each variable at every node of the tree by propagating local consistency properties in the constraint network. In our algorithm, the search tree is explored by following the *weighted degree* [Boussemart et al., 2004] adaptive heuristic which learns from the failures during the search, so that the variables involved in the constraints that have been violated the most so far are instantiated first. This heuristic proved to be particularly efficient on this problem, as it dynamically focuses on the hardest parts of the CSP first.

Optimization. The optimization criterion c simply is the sum of the costs of each single maneuver as defined in equation 3. The optimization algorithm used to solve the CSP is an adaptation of the backtracking algorithm called *branch and bound*: each time a solution with cost c_s is found, the constraint $c < c_s$ is dynamically added to the CP model, and the search is resumed to look for a better solution.

Eventually, the search for a better solution will fail, *proving* that the best solution so far was optimal (or, if no solution has been previously found, that there is no solution satisfying all the constraints). To quickly obtain solutions of good quality, which is mandatory in an operational real-time context, our search strategy first focused on maneuvers that least increase the cost.

RESULTS

The benchmark generation and the two conflict resolution algorithms were implemented, using the FaCiLe [Barnier and Brisset, 2001] constraint library for the CP model. The following results were obtained on a standard workstation consisting of an octo-core Intel® Xeon® processor running at 3.4 GHz and equipped with 8 GB of memory.

Benchmark

A total of 120 instances were produced, based on situations with 5, 10, 15 and 20 aircraft in the same airspace volume and with uncertainty levels 1, 2 and 3, thus changing the density of the problem. Ten

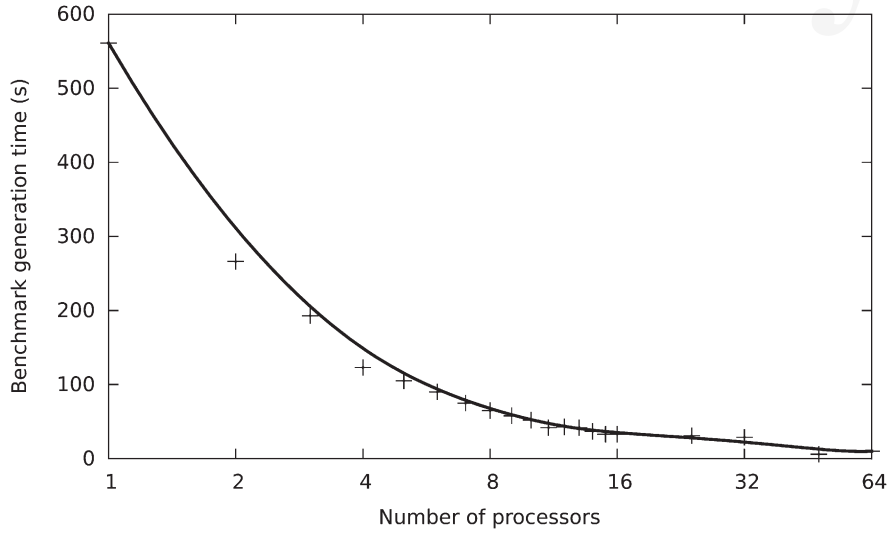

Figure 6. Benchmark generation time w.r.t. number of processors. The horizontal scale is logarithmic.

random instances were created for each set of parameters, in order to assess the reliability of the resolution algorithms.

The generation of a given instance is highly parallelizable (the computation of the constraint between two given aircraft is independent from other constraints in the problem), which made it possible to dramatically reduce the needed computation time. As an example, the biggest and hardest instances (20 aircraft with high uncertainty level) were produced in less than three minutes while the smallest ones only needed a few seconds.

Figure 6 shows the influence of the number of processors used on F6 the benchmark generation time for a given instance. The time saving is quite huge, since only 10 seconds are necessary with 64 processors where it took more than 9 minutes to a single processor. The gain, however, becomes less interesting when the number of processors further increases, because the communication overhead between processes then takes a significant amount of time. For the type of instances we generated, 16 processors seemed to constitute a fair compromise.

Conflict Resolution

The resolution algorithms were both limited to a 5 minutes execution time, to be compatible with the time constraints of an operational setting. In this context, all feasible instances were solved within seconds, and an optimality proof was obtained for most of them. Figure 7 shows a solution for a 10-aircraft conflict. F7

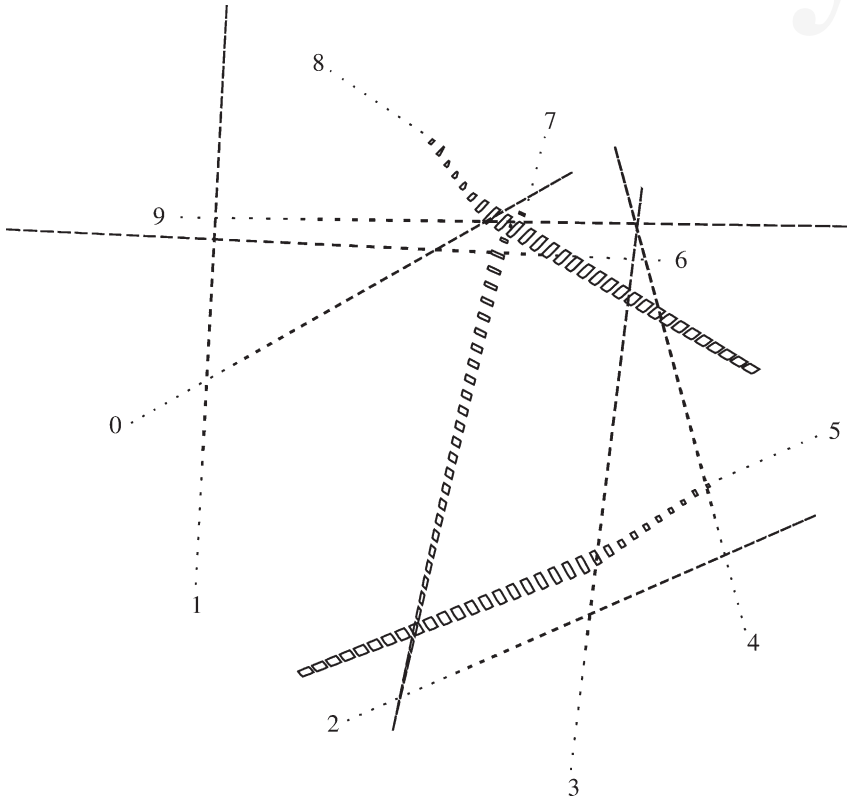

Figure 7. A solution to a 10-aircraft conflict. Trajectories are depicted as sequences of convex hulls, representing the uncertainty.

Computing Times. In more details, table 1 provides the computation times (averaged over the 10 different instances for each set of parameters) for finding the best solution. Instances with 5 and 10 aircraft are efficiently solved (under one second) by both algorithms (CP being a bit faster than EA). Most 15-aircraft instances are solved within one minute, while 20-aircraft instances often need a few minutes. Moreover, a proof of optimality is obtained (with CP only) on all instances with 5 and 10 aircraft and almost all instances with

Table 1. Average time (in seconds) for Finding Best Solution with EA and CP Algorithms for Each Set of Parameters

	n							
	5		10		15		20	
	CP	EA	CP	EA	CP	EA	CP	EA
ε_{low}	0.00	0.02	0.22	0.97	24.08	2.01	75.14	95.98
ε_{med}	0.00	0.02	0.27	1.44	45.17	32.60	79.61	184.61
$\varepsilon_{\text{high}}$	0.00	0.02	1.04	0.37	48.59	93.19	58.44	274.16

15 aircraft. When 20 aircraft are involved, however, optimality proof is not reached within the five minutes time limit.

Particularly interesting is the fact that, for instances that do not have any solution, a proof of non-feasibility is obtained within one second. This could make it possible to generate, in a real-time setting, a new instance where, for the same situation, more maneuvers would be allowed, hopefully giving a resolution to the conflicting situation.

Finally, in almost all instances, including the toughest ones, a first solution was found within seconds. This means that in a real-time operational context, it could be possible to provide the controllers quickly with a first set of maneuvers that solves the conflict, so that it could be their choice to transmit them right away or wait for a more efficient solution, depending on their current workload and the urgency of the situation.

Cost of Solutions. Table 2 provides average costs for each set of parameters. According to the definition given in equation 2 each maneuver has a cost belonging to the interval [0, 50] for the investigated instances. As expected, the cost increases with the number of aircraft involved, because the density of aircraft and conflicts increases with this parameter for a given constant airspace volume. The maneuver cost per aircraft varies from less than 1 for the smallest instances to 15 for the hardest ones.

Figure 8 depicts the cost of the best solution found with respect to the *intrinsic difficulty* ρ of the instance. The intrinsic difficulty is here defined as the total number of forbidden couples of maneuvers:

$$\rho = \sum_{\substack{i,j \in [1,n]^2 \\ i<j}} |c_{ij}|$$

where c_{ij} is the constraint between aircraft i and j, as defined in equation 4. Clearly, the cost of the best solutions is closely correlated to the intrinsic difficulty of the problem, which could be used a priori to determine the expected efficiency of resolution.

Table 2. Average cost of best solutions for each set of parameters. 2-in-1 cells correspond to sets of parameters where both CP and EA reached optimal solution. For $n = 15$ and $n = 20$, results include solutions that were not proved optimal

	n							
	5		10		15		20	
	CP	EA	CP	EA	CP	EA	CP	EA
ε_{low}	5.3		29.8		86.3	86.8	185.8	176.9
ε_{med}	4.2		46.6		104.0	104.0	267.6	282.8
$\varepsilon_{\text{high}}$	5.1		45.7		170.4	156.3	299.0	305.0

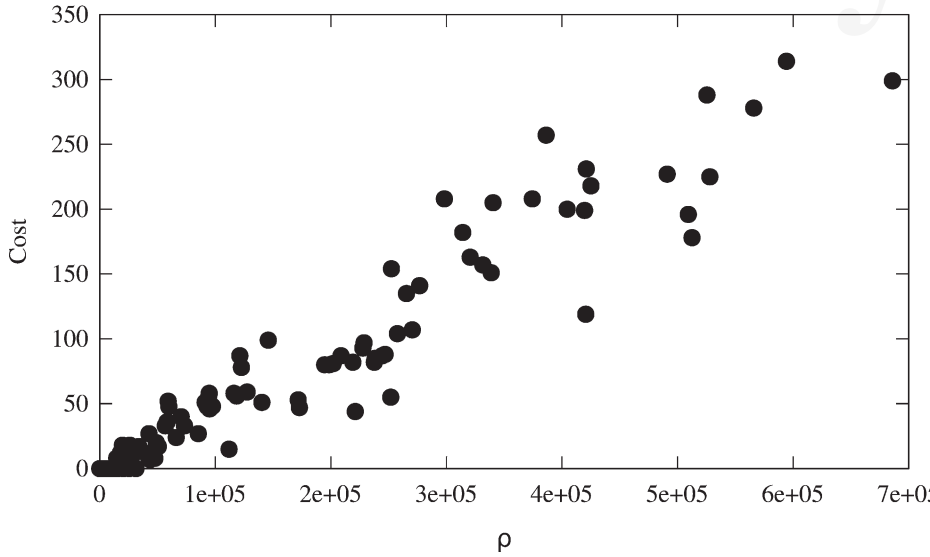

Figure 8. Cost of the best solution found w.r.t. the intrinsic difficulty ρ of the instance.

In terms of cost, CP and EA are equivalently efficient: they both reach optimal solutions for almost all instances involving 15 or less aircraft, and alternately give the best solution for 20-aircraft instances. It would therefore be interesting to run both algorithms in parallel for a given instance, to get the best possible solution.

CONCLUSION AND FURTHER WORK

We have presented a new benchmark framework for generating and solving air traffic conflict resolution problems with many configuration opportunities. Unlike other previous approaches, we have proposed to separate the generation of instances from their resolution, giving the possibility to easily test different algorithms for solution search and optimization.

The production of the benchmark is highly configurable: the density of the conflict (controlled by the number of aircraft involved or the volume of the considered airspace), the number of authorized maneuvers and the level of uncertainty to be taken into account are the main parameters, but the tuning can be even finer, e.g., with the possibility of defining custom maneuvers or trajectory uncertainties. The output is a data file containing all pre-computed trajectories and a list of maneuvers pairs that cannot be performed simultaneously. As this phase is highly parallelizable, this method can be used to generate an entire benchmark database within a reasonable computation time.

To illustrate the usefulness of our benchmark framework, we have also described two different approaches to solve the generated conflict problems, an Evolutionary Algorithm and a Constraint Program, and have shown how to compare their results fairly on the 120 instances of various difficulties of our proposed benchmark basis. Most of these instances were solved in less than one second, the hardest ones needing a few minutes of computation. With the CP algorithm, optimality proofs were obtained in most cases, and instances without solution were proved inconsistent within one second. As expected, the cost of the solutions, i.e., the sum of maneuver costs defined in the conflict data, increases with the intrinsic difficulty of the instance, defined as the overall amount of forbidden maneuver pairs.

We plan to extend our approach to consider vertical maneuvers, like a flight level change, interrupted climb or anticipated descent, thus increasing the configurability and generality of the framework. In terms of efficiency, the detection phase could be enhanced by the use of a fastest algorithm for computing distances between the convex hulls that model the uncertainties.

The realism of the instances can be greatly improved by integrating the conflict generation into our fast-time simulation platform CATS (or other third-party simulators), to extract the conflicting situations from the simulated traffic. This would also make it possible to test resolution algorithms in a fast-time simulation setting over a whole day of traffic.

Finally, we are currently working on yet other algorithms for the conflict resolution problem, such as an ad hoc *branch and bound* and a *Tabu Search*, and their hybridization to increase the efficiency and the robustness of the resolution.

ACRONYMS

FMS	Flight Management Systems
BADA	Base of Aircraft Data
EA	Evolutionary Algorithm
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
CATS	Complete Air Traffic Simulator

REFERENCES

Alliot, J.-M., Gruber, H., and Schoenauer, M. (1993). Using genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE.

Alonso-Ayuso, A., Escudero, L., and Martin-Campo, F. (2011). Collision avoidance in air traffic management: a mixed-integer linear optimization approach. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):47–57.

Barnier, N. and Allignol, C. (2012). Trajectory deconfliction with constraint Programming. *The Knowledge Engineering Review*, 27(03):291–307.

Barnier, N. and Brisset, P. (2001). FaCiLe: a Functional Constraint Library. In *Colloquium on Implementation of Constraint and LOgic Programming Systems CICLOPS'01 (Workshop of CP'01)*, Paphos, Cyprus.

Boussemart, F., Hemery, F., Lecoutre, C., and Sais, L. (2004). Boosting systematic search by weighting constraints. In *European Conference on Artificial Intelligence ECAI*, pages 146–150.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (1998). *Computational Geometry – Algorithms and Applications*. Springer, second edition.

Delahaye, D., Peyronne, C., Mongeau, M., and Puechmorel, S. (2010). Aircraft conflict resolution by genetic algorithm and B-spline approximation. In *ENRI International Workshop on ATM/CNS*, Tokyo, Japan. EIWAC.

Durand, N. (2004). *Algorithmes Gntiques et autres mthodes d'optimisation appliqus la gestion de trafic arien*. Thse d'habilitation, cole Doctorale d'Informatique de Toulouse.

Durand, N., Alliot, J.-M., and Noailles, J. (1994). Algorithmes génétiques : un croisement pour les problémes partiellement séparables. In *Proceedings of the Journées Évolution Artificielle Francophones*. EAF.

Durand, N., Alliot, J.-M., and Noailles, J. (1996). Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM.

Durand, N. and Granger, G. (2003). A traffic complexity approach through cluster analysis. In *5th ATM R&D Seminar*.

Erzberger, H. (1997). Conflict probing and resolution in the presence of errors. In *Procceedings of the 1st USA/Europe ATM R&D Seminar*.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Graham, R. L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. In *Information Processing Letters*.

Granger, G., Durand, N., and Alliot, J. (2001). Optimal resolution of en route conflicts. In *4th ATM R&D Seminar*.

Michalewicz, Z. (1992). *Genetic algorithms + Data Structures = Evolution Programs*. Springer-verlag.

Pallottino, L., Féron, E., and Bicchi, A. (2002). Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):3–11.

Rey, D., Rapine, C., Fondacci, R., and Faouzi, N. E. (2012). Minimization of potential air conflicts through speed regulation. *Transportation Research Record: Journal of the Transportation Research Board*, 2300:59–67.

Van Hentenryck, P. (1995). Constraint solving for combinatorial search problems: a tutorial. In Montanari, U. and Rossi, F., editors, *Principle and Practice of Constraint Programming CP'95*, volume 976 of *Lecture Notes in Computer Science*, pages 564–587, Cassis, France.

Vela, A., Solak, S., Singhose, W., and Clarke, J. (2009). A mixed integer program for flight-level assignment and speed control for conflict resolution. In *Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*. IEEE.

Yin, X. and Germay, N. (1993). A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In Albrecht, R. F., Reeves, C. R., and Steele, N. C., editors, *Proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference*, Innsbruck, Austria. Springer-Verlag.

Zeghal, K. (1993). Techniques réactives pour l'évitement. Technical report, ONERA.

BIOGRAPHIES

Cyril Allignol is an assistant professor at École Nationale de l'Aviation Civile (ENAC). He graduated from ENAC as an engineer in 2006, and received a Ph.D. (2011) in computer science from the University of Toulouse.

Nicolas Barnier is an assistant professor 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.

Nicolas Durand graduated from the École Polytechnique de Paris in 1990 and ENAC in 1992. He has been a design engineer at the Centre d'Études de la Navigation Aérienne (then DSNA/DTI R&D) since 1992, holds a Ph.D. in Computer Science (1996) and got his HDR (French tenure) in 2004. He is currently professor at the ENAC/MAIAA lab.

Jean-Marc Alliot graduated from the École Polytechnique de Paris in 1986 and ENAC in 1988. He received a Ph.D. in 1992 in Computer Science and got his HDR (French tenure) in 1996. He has been the head of LOG (Global Optimization Lab, ENAC/CENA) and then the head of DSNA/DTI R&D until 2011. He is currently research director at IRIT (Institut de Recherche en Informatique de Toulouse).