Large Scale 3D En-Route Conflict Resolution

Cyril Allignol, Nicolas Barnier, Nicolas Durand, Alexandre Gondran, Ruixin Wang

Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)

Large-Scale 3D En-Route Conflict Resolution

Cyril Allignol, Nicolas Barnier, Nicolas Durand, Alexandre Gondran, Ruixin Wang

ENAC
7 av. Édouard Belin
31055 Toulouse Cedex, France
allignol,barnier,durand,gondran,ruixin.wang@recherche.enac.fr

***Abstract*—En-route conflict resolution is a good example of a large-scale combinatorial optimization problem. On the one hand, it has been modeled in many different ways, most of the time depending on the tools that were proposed to solve it. On the other hand, many different resolution methods can be tested and compared on such problems but a common model needs to be used to validate the comparison.**

In this paper we extend the 2D-framework introduced in 2013, which separates the model from the solver. First, we introduce a 3D-model and add new refinements on the uncertainty model taking into account, inter alia, delays due to human factors. Second, we compare the performance of a complete Constraint Programming solver and an approximation algorithm based on a Memetic Algorithm, an efficient metaheuristic combining Genetic Algorithm with Tabu Search.

To this aim, we generate a benchmark of conflict resolution problems built with scenarios involving 15 to 100 aircraft, 3 different levels of uncertainty and maneuvers in both horizontal and vertical planes. The two methods are able to efficiently solve moderate size problems in near real time, but the execution time of the complete algorithm exponentially rockets with larger instances whereas the metaheuristic scales much better with the number of aircraft. However, the former is able to prove optimality or infeasibility on reasonable problems, which allows the assessment of the quality of the solutions produced by the latter.

Keywords: conflict resolution, Metaheuristics, constraint programming

INTRODUCTION

Research on automatic conflict resolution started in the 1980s and many different models were introduced to comply with existing resolution techniques. Some research, issued from the Air Navigation System Providers offer realistic models [1] but do not focus on the resolution methods. Some approaches, like [2], [3] which use uncertainty models and the Base of Aircraft Data (BADA, developed and maintained by Eurocontrol), focused both on the model and the resolution algorithms. However, they were completely tailored to the underlying traffic simulator (CATS), which prevents the scientific community from comparing different resolution methods.

Many mathematical models have led to specific resolution algorithms able to deal with very complex situations, but require specific characteristics for trajectory prediction. This is the case for Pallottino's approach [4] that used Mixed Integer Linear Programming (as [5], [6], [7]). Theses models rely on constant speed trajectories and assume that all maneuvers are executed simultaneously. They cannot deal with trajectory models able to handle descending or climbing aircraft, nor with complex trajectory uncertainties.

Conflict resolution is known for being highly combinatorial [8] and large instances can therefore be very difficult to solve. Assessing the relative merits of different solvers is very useful to pave the way to future automation tools.

In 2013, we proposed a framework to separate the trajectory model from the resolution algorithm [9] and offered the scientific community the opportunity to test different algorithms on various problems without investing efforts on the model. With such a framework, resolution times and costs of different solvers can be fairly compared. Problems can be easily downloaded on a website clusters.recherche.enac.fr. For each problem, the website offers a second file containing the trajectory envelops in order to be able to visualize the solutions. These problems were solved using an Evolutionary Algorithm and Constraint Programming, which were able to find the best solution on small problems or highly constrained ones.

In 2015, Lehouiller et al. [10] also proposed a general framework by modeling the problem with a graph where the vertices are the trajectories and the edges connect compatible trajectories. This is possible because the problem only involves binary constraints. The problem can thus be viewed as minimizing the cost of a maximum clique. In figure 1, each aircraft must choose between four maneuvers $\{a, b, c, d\}$. The edges represent compatible maneuvers. There is only one maximum clique representing a solution: $< 1d, 2c, 3c, 4b >$.

Lehouiller obtains good results using this model on problems involving up to 20 aircraft with a small number of maneuver options. Lehouiller's graph model can be generated with the 2013 framework.

In this article, we try to build more realistic examples using vertical maneuvers as well as horizontal ones. We also improve the uncertainty model to make it compliant to air trafic controller and pilot communication constraints and practices. For example, when a pilot is given a heading or flight level change, it may take some time before the maneuver is actually implemented. When the aircraft heads back to its initial trajectory, an extra execution time also needs to be taken into account. Heading changes are not always followed with high precision, and some level of uncertainty should be added to any trajectory with this degree of freedom. The speed error needs to be taken into account in the model, as well as uncertainties on the climbing rates in a 3D context. Moreover, an aircraft can either "fly by" or "fly over" a beacon, which

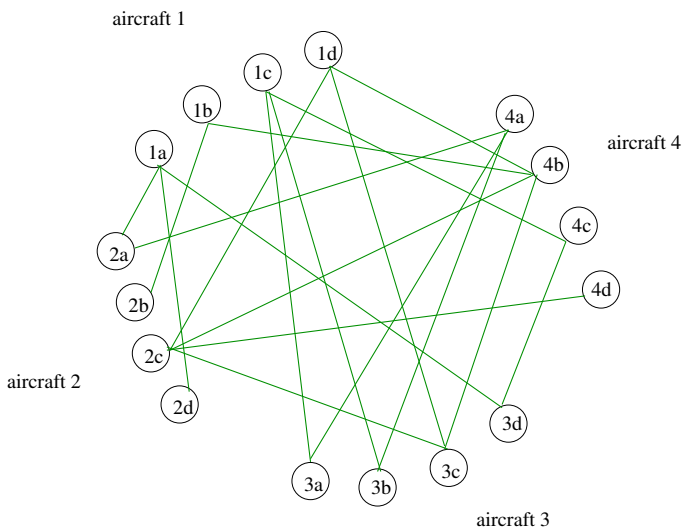

Fig. 1. Graph representation of a Conflict Resolution Problem: Clique $< 1d, 2c, 3c, 4b >$ is a solution.

creates an extra source of uncertainty. The size of the 3D instances are also much larger as they involve aircraft on different flight levels, which allows us to check the limits of solvers on very difficult problems.

The first section of this paper introduces the improved trajectory model taking into account vertical maneuvers and realistic uncertainties. Section III details the method used to build conflict resolution benchmarks with different sizes and levels of uncertainties. Section IV decribes two methods for the resolution of conflicts, namely Constraint Programming and Memetic Algorithm. Section V compares both approaches on the benchmark and assesses their comfort zones.

I. Trajectory Prediction Model

In this paper, we build a trajectory prediction tool giving the aircraft positions at each time step according to simple maneuver options, and a set of uncertainty parameters. In our previous work, we had only defined heading change maneuvers in order to comply with controllers practices in the horizontal plane. In this article we add vertical maneuvers in order to ease the resolution of complex situations. We also try to be more realistic in the trajectory model by taking aircraft turning rates into account and uncertainties related to human pilot and controller behaviors. In the end, we compute up to 64 different trajectories for each maneuver taking into account the uncertainty parameters. The 3D convex envelops of the trajectories are built and compared to other aircraft trajectories to detect possible conflicts.

A. Maneuvers

We discretize time into steps of duration τ to describe maneuvers. τ is small enough to detect every conflict in the application. In section III, $\tau = 3\,\mathrm{s}$ because two facing aircraft flying at $600\,\mathrm{kn}$ (maximal speed) get only $1\,\mathrm{NM}$ closer every $3\,\mathrm{s}$, so we miss no conflict with such a small τ value (see [11] for a more in-depth discussion on this topic).

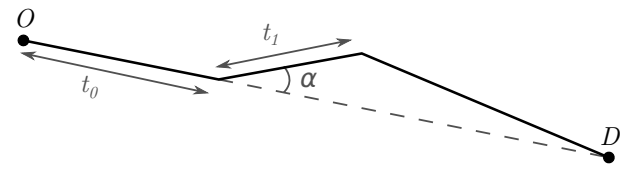

Fig. 2. Maneuver model.

Aircraft fly on five different levels in the vertical plane, from Flight Level (FL) 280 to FL320. We do not model climbing or descending flights but it could easily be done. On each Flight Level, flight plan routes are defined by a list of beacons. The first point O is the origin and the last point D is the destination. Aircraft fly from beacon to beacon and correct the lateral error thanks to their FMS. Consequently, when heading toward a beacon, there is no uncertainty on the lateral position.

In our trajectory model, maneuvers are either heading changes or flight level changes. We do not combine both in order to keep maneuvers simple. They are started at a time t_0 and ended at another time t_1. Heading changes α can take $n_\alpha = 6$ different values in our benchmark, i.e. 10, 20 or 30 degrees to the left or the right of the current heading, vertical moves δ_{FL} can take $n_{\mathrm{FL}} = 4$ values, i.e. climb or descend $1000\,\mathrm{ft}$ or $2000\,\mathrm{ft}$ from the current level. The number of maneuver kinds is thus $n_k = n_\alpha + n_{\mathrm{FL}} = 10$. We limit the number of maneuvers created by choosing t_0 among n_0 values (typically $n_0 = 4$ in the experimental benchmark). The number of values for t_1 is also chosen among a limited set of n_1 values (typically $n_1 = 4$).

Horizontally, a heading change maneuver, as depicted in figure 2, is representative of current Air Traffic Control practice and can be easily implemented by pilots and current FMS technologies (cf. [3]).

If we consider 4 values for t_0, 4 values for t_1, 4 vertical maneuvers and 6 possible angles (there is no use to combine a null heading change $\alpha = 0$ with various t_0 and t_1 values, so that only one maneuver is added when the aircraft is not deviated), the number of maneuvers per aircraft is:

$$n_{\mathrm{man}} = n_0 \times n_1 \times n_k + 1$$

So for the benchmark presented in section III $n_{\mathrm{man}} = 4 \times 4 \times 10 + 1 = 161$.

For a n aircraft conflict, the search space size is n_{man}^n, i.e. $\approx 10^{44}$ for a 20-aircraft instance (almost 5.10^{220} for 100 aircraft).

We model six different sources of uncertainties:

- When a pilot gets a maneuver order, she can react more or less quickly. An uncertainty $\varepsilon_{t_0} \in [0, E_{t_0}]$ representing the maximum reaction time for beginning a maneuver is associated to time t_0 (see figure 3).
- An uncertainty $\varepsilon_{t_1} \in [0, E_{t_1}]$ representing the maximum reaction time for ending a maneuver is associated to time t_1 (see figure 3).
- An uncertainty $\varepsilon_\alpha \in [-E_\alpha, E_\alpha]$ is also associated to the heading change angle α (see figure 4).
- Horizontal aircraft speeds v_h are hence subject to a $\varepsilon_{v_h} \in [-E_{v_h}, E_{v_h}]$ relative error (expressed as a percentage)

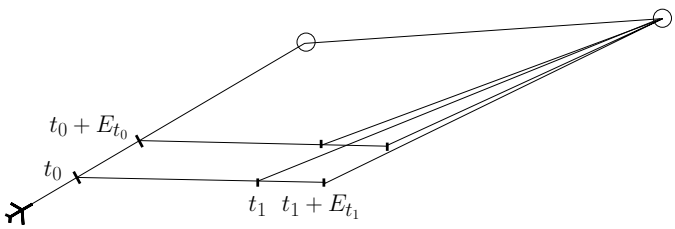

Fig. 3. Reaction time uncertainty model with maximal errors E_{t_0} and E_{t_1}.

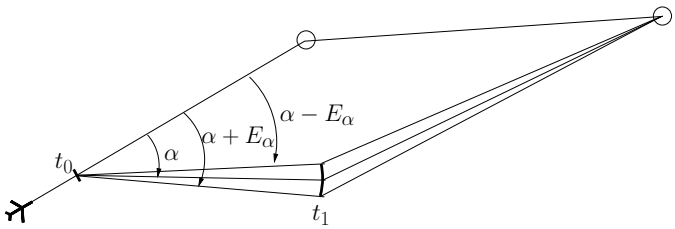

Fig. 4. Heading change uncertainty model with maximal error E_α.

such that future positions of aircraft are spread over a range which grows with time (see figure 5).

- Climbing and descending rates v_v are also subject to a $\varepsilon_{v_v} \in [-E_{v_v}, E_{v_v}]$ relative error (as a percentage) (see figure 6).
- An aircraft can "fly by" (F_b) or "fly over" (F_o) a beacon, and we consider both options to build the future trajectory (see figure 7). The fly mode f_m can be chosen among two values $f_m \in \{F_b, F_o\}$.

B. Decision Variables

We simplify the access to the conflict matrix C: the decision variables t_0, t_1 and the heading change maneuver α or the flight level change δ_{FL} associated with aircraft i are aggregated into a single decision variable m_i by a bijection from the valid 4-tuples to interval $[1, n_{\text{man}}]$. We call M the set of decision variables of the problem:

$$M = \{m_i \in [1, n_{\text{man}}], \forall i \in [1, n]\} \tag{1}$$

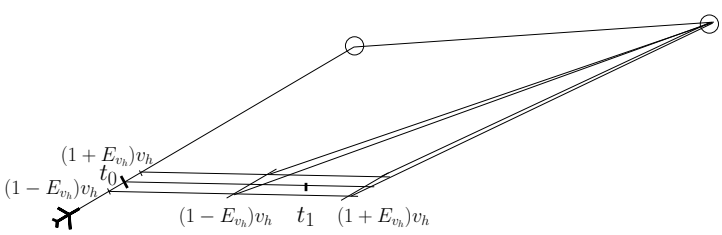

Fig. 5. Speed uncertainty model with maximal error ε_s.

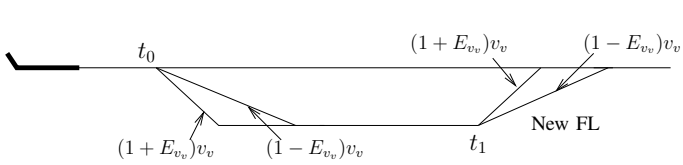

Fig. 6. Climb and descent uncertainty model with maximal error E_{v_v} model.

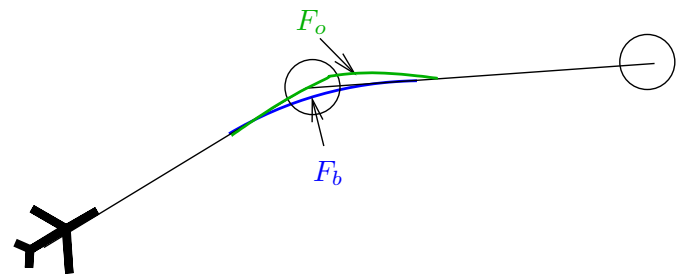

Fig. 7. Flight mode uncertainty model with possible modes "fly by" (F_b) or "fly over" (F_O).

C. Cost

We compute the maneuver cost of our model from the decision variables. Values of t_0 are enumerated by an index k_0 varying in $[1, n_0]$, values of t_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 $[1, \frac{n_\alpha}{2}]$. Flight levels are indexed by k_v in $[1, \frac{n_{\text{FL}}}{2}]$: for a 1000 ft vertical move $k_v = 1$, for a 2000 ft move $k_v = 2$. For our benchmark problems, the cost $c(m_i)$ of a maneuver m_i for aircraft i is then defined by:

$$c(m_i) = \begin{cases} (n_0 - k_0)^2 + k_1^2 + k_\alpha^2 & \text{if } \alpha \neq 0 \\ (n_0 - k_0)^2 + k_1^2 + (1 + k_v)^2 & \text{if } \delta_{\text{FL}} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where k_0, k_1, k_α and k_v are the indices 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;
5) vertical maneuvers should be as small as possible;
6) a 1000 ft vertical maneuver is equivalent to a 20° heading change, and a 2000 ft one to 30°.

We chose to keep the expression of the cost function very simple in order to make it easy to understand. In a real environment, it should be modified to comply with aircraft performance models on one side and controllers' preferences on the other side. This paper aims at specifying a framework that separates the solver method from the problem itself, so as to provide the scientific community (which may be unfamiliar with ATC and conflict resolution) with the simplest possible benchmark that enables them to compare different solvers.

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

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

D. Handling Uncertainties

We compute the envelops for each maneuver in order to detect conflicts between two maneuvers of two different aircraft, while taking the various uncertainties into account.

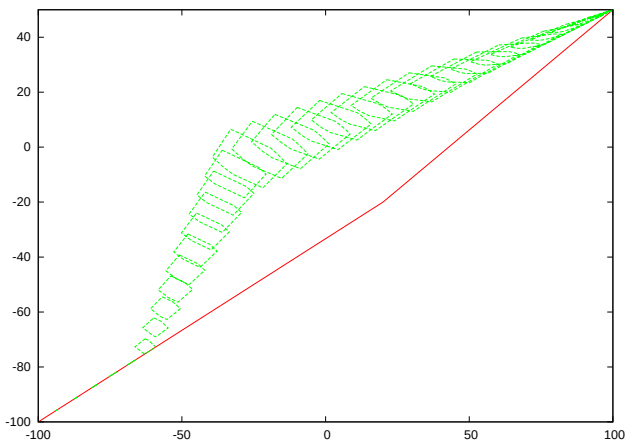

Fig. 8. An example of convex hulls representing a maneuver with uncertainties in the horizontal plane.

Maneuver descriptions are stored in a table that defines, for each aircraft and each maneuver, the possible future positions of the aircraft at every time step. These positions are safely approximated horizontally by their convex hull, which is computed with Graham's algorithm [12], and vertically by a minimum and maximum altitude.

As described in section I-A, we defined six uncertainty parameters: ε_{t_0}, ε_{t_1}, ε_α, ε_{v_h}, ε_{v_v} and f_m. In order to take into account every possible trajectorie, we test every combination of the extreme values of the uncertainties:

1) 0 and E_{t_0} for ε_{t_0};
2) 0 and E_{t_1} for ε_{t_1};
3) $-E_\alpha$ and E_α for ε_α;
4) $-E_{v_h}$ and E_{v_h} for ε_{v_h};
5) $-E_{v_v}$ and E_{v_v} for ε_{v_v};
6) F_b and F_o for f_m.

$2^6 = 64$ combinations of the extreme uncertainty values are used to build 64 trajectories. Once these trajectories are built, we use Graham's algorithm to build the convex hull of the 64 extreme trajectories for each time step t of the trajectory. This convex hull represents the possible positions of the aircraft at every time step t.

Figure 8 gives an example of a 30° heading change maneuver starting at $t = 5\,\text{min}$, lasting $10\,\text{min}$, with $E_{t_0} = E_{t_1} = 60\,\text{s}$, $E_\alpha = 5°$ and $E_{v_h} = 5\,\%$. The convex hulls including the possible aircraft positions every minute are represented with a green dashed line.

Any traffic simulator using any kind of uncertainty hypothesis can be adapted to build the trajectory prediction for the aircraft and for the maneuver options. It is also possible to have different uncertainties for different aircraft. We can also use different maneuver options if necessary. The convex hull prevents the detection algorithm from missing a potential conflict w.r.t. the chosen uncertainties.

II. Conflict Detection

Once the trajectory predictions are computed and stored, we can build the 4D conflict matrix C. We combine the four maneuver parameters t_0, t_1, α and δ_{FL} in order to reduce the 4-tuple to a single number. Therefore we use a bijection from the valid 4-tuples 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 are in conflict. In this case, $C_{i,j,k,l} = 1$, otherwise $C_{i,j,k,l} = 0$. Furthermore, we can consider that $i < j$ since a conflict between i and j is equivalent to a conflict between j and i. To detect a conflict, the minimal vertical distance between the convex hull pairs representing aircraft i and j is first computed at each time step and compared to the vertical standard separation norm (1000 ft), then, if the norm is violated, the horizontal distance between the two convex hulls is computed and compared to the horizontal standard separation norm (5 NM).

For each time step, the algorithm is divided in four stages:

1) Check the vertical minimum distance between convex hulls.
2) 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.
3) Otherwise, check if two edges of convex hulls k and l intersect.
4) 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 time consuming because the number of pairs tested is big. For example, a 20-aircraft conflict with 161 maneuvers per aircraft generates $\frac{20\times 19}{2} = 190$ pairs of aircraft for which $161^2 = 25,921$ pairs of maneuvers must be tested. A total of $4,924,990$ pairs of maneuvers must be tested to build the conflict matrix.

III. Benchmark Generation

In section V, we consider 11 sizes of instances, involving 15, 20, 25, 30, 40, 50, 60, 70, 80, 90 and 100 aircraft, with three levels of uncertainties ($\varepsilon \in \{1, 2, 3\}$). For each combination, 10 scenarios of aircraft converging to the center of the considered airspace volume were randomly built. For each scenario, speeds are chosen from 384 kn to 576 kn (i.e. 20 % variation almost a typical speed of 480 kn). The nominal vertical speed for maneuvers is set to $600\,\text{ft}\,\text{min}^{-1}$. The initial aircraft positions are chosen on a 100 NM radius circle and are noised within a 20 NM-side square. The initial heading is also noised with a value chosen in $[-1, 1]$ radians ($\approx \pm 60°$). In the vertical plane, aircraft are equally dispatched on 5 different levels from $FL280$ to $FL320$. A total of 11×10 scenarios are built. Figure 9 illustrates these dimensions.

For the maneuvers, we choose α values in the set: $\{-30°, -20°, -10°, 10°, 20°, 30°\}$. Flight level changes δ_{FL} are chosen in the set $\{+10, +20, -10, -20\}$. t_0 can take four values: 0,1,2 or 3 minutes. t_1 can take four values: $t_1 = t_0 + s$

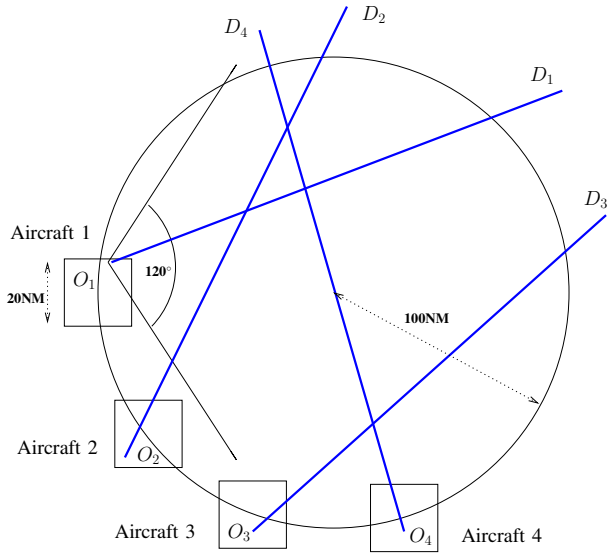

Fig. 9. Geometry of conflict scenario generation.

Uncertainty level	$\varepsilon = 1$	$\varepsilon = 2$	$\varepsilon = 3$
E_{t_0}	10 s	20 s	30 s
E_{t_1}	10 s	20 s	30 s
E_α	1°	2°	3°
E_{v_h}	2 %	4 %	6 %
E_{v_v}	5 %	10 %	15 %
f_m	$\{F_b, F_o\}$	$\{F_b, F_o\}$	$\{F_b, F_o\}$

TABLE I
UNCERTAINTY PARAMETERS

with s chosen among 5,6,7,or 8 minutes for heading changes and 7,8,9 or 10 minutes for flight level changes.

For each scenario, three levels of uncertainty are defined. The uncertainty parameters are summarized in table I. The size of the convex hulls approximating the possible positions of aircraft increases with the uncertainty parameters, creating more conflicts for the same scenario. A total of $11 \times 3 \times 10 = 330$ scenarios were thus built and tested with two different approaches in the next section.

IV. Conflict Resolution

In this section, we propose two methods for the resolution of the generated conflicts. The first one, a Memetic Algorithm (section IV-A), is a metaheuristic that mimics natural evolution to explore the search space. The second one, Constraint Programming (section IV-B), is based on a systematic exploration of the search space, which can prove the optimality (or the absence) of a solution.

A. Memetic Algorithm

1) Principles: The Memetic Algorithm (MA), described in algorithm 1, is an hybridization of an Evolutionary Algorithm (EA) and a Tabu Search (TS) such as presented in [13]. The main feature of a MA is that each element of the population is a local minimum.

Algorithm 1 Memetic algorithm (MA)

1: population ← initializePopulation()
2: **while** termination criterion is not met **do**
3: parent$_1$, parent$_2$ ← select(population)
4: child ← crossover(parent$_1$, parent$_2$)
5: child ← tabuSearch(child)
6: population ← replace(population, child)
7: **end while**
8: Return the best element of population

First (line 1), a population of candidate solutions in the state space is randomly generated and a tabu search (described in algorithm 2) is applied to each candidate. In a second step (line 3) we randomly select two individuals in the population called *parents*. Then (line 4), we generate a new element called *child* by a classical *crossover* operator of the two parents, which recombines their maneuvers in the offspring. Afterwards (line 5), we improve the child by applying a tabu search. Finally (line 6), we replace the worst element of the population by the child if its cost is lower and if it does not already belong to the population. We iterate this procedure until a given time limit (line 2) or when no improvement is made for a given number of iterations.

2) Fitness Function: The fitness of our MA represents the function to minimize. Here, we first focus on finding a conflict-free set of maneuvers, with the smallest deviation cost as a secondary objective. Therefore, the fitness function is defined as the linear combination of two terms, the number of remaining conflicts and the cost of a solution defined by equation 3:

$$F = M \times \sum_{i<j} C_{i,j,m_i,m_j} + \text{cost}$$

where M is a big (enough) integer to guarantee that the fitness of a solution with more conflicts is always higher.

3) Tabu search: Our tabu search algorithm, described in algorithm 2, improves the current solution according to the fitness function. A TS, like all local search algorithms, modifies the current solution by successive small changes called *moves*. The moves used in our TS consist in the modification of the maneuver assigned to one of the aircraft, which defines the *neighborhood* of a candidate solution.

Algorithm 2 Tabu Search (TS)

1: **Input :** an initial solution, s
2: tabuList ← ∅
3: **while** termination criterion is not met **do**
4: mv ← selectBestMove(s, tabuList)
5: s ← move(s, mv)
6: tabuList ← update(tabuList, reverse mv)
7: s_{best} ← saveBest(s_{best},s)
8: **end while**
9: Return s_{best}

First (line 4), we select the best move mv for the current solution which is not in *tabuList*, i.e. the list of forbidden

maneuvers for each aircraft during a given number of iterations, being initially empty (line 2). Notice that in a TS, the selected move is not necessarily a move that improves the solution fitness. In a second step (line 5), we update the current solution by applying the selected move and add to *tabuList* its reverse (line 6) to prevent the change to be undone during a given amount of iterations. Finally (line 7), we register the current solution as the best solution encountered so far if it is the case. We iterate this procedure until a given number of iterations (line 3).

4) Crossover: We have used a very basic crossover operator that generates a child solution from two parent solutions. For each aircraft, the crossover operator randomly selects one of the two maneuvers of the parents and assigns it to the child.

B. Constraint Programming

Constraint Programming (CP) is a versatile optimization technology based on the Constraint Satisfaction Problem (CSP) formalism which 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 quick development of solvers in an incremental fashion and experimentation with various search strategies without changing the model. See [14] for example, for more details on the CP technology.

Contrar to metaheuristics, the standard search algorithm of CP is *backtracking* (or *Branch & Bound* if a cost is to be optimized), which provides a complete algorithm able to prove the optimality of a solution or the infeasibility of an instance. These properties allow us to assess the efficiency of the MA, which is able to consistently find the optimal solution for all instances for which the constraint program was able to obtain proof of optimality (up to 30 or 40 aircraft as described in section V). However, the computation times of our constraint program become prohibitive for the largest instances, because the search algorithm has an exponential complexity w.r.t. the number of aircraft.

We use here the CP algorithm described in [9] with the same decision variables as the MA and standard binary constraints to represent the conflicts between maneuvers. Because the solutions produced by constraint programming necessarily satisfies all constraints, the cost is only defined by the sum of the costs of all aircraft as defined by equation 3.

V. Results

Benchmark generation (section III) and conflict resolution (section IV) were implemented, using the FaCiLe constraint library [15] for the CP model. The following results were obtained on a standard workstation consisting of a 3.4 GHz octo-core Intel® Xeon® processor with 16 GB of memory running Debian Linux 3.16.7. The instances used in this study are available online at clusters.recherche.enac.fr.

To assess the performances of the MA, which is a stochastic algorithm using a pseudo-random number generator, 5 runs were attempted for each instance to investigate the robustness of the algorithm. All tests were done with a population of 40 individuals with 1000 iterations for the tabu search phase and an overall time limit of 300 s.

Compared to the benchmark presented in [9], ranging from 5 to 20 aircraft and limited to maneuvers in the horizontal plane, this new benchmark is an order of magnitude greater with 3D instances from 15 to 100 aircraft evenly distributed among 5 flight levels (cf. section III). Therefore the mean number of aircraft on each level ranges from 3 (for 15-aircraft instances) to 20 (for 100-aircraft instances), which approximately amounts to the instances of the 2013 benchmark for a single layer. However, as an aircraft may climb or descend and interferes with the other levels, the various layers are obviously not independent and the size of the search space (exponentially) increases.

The next sections detail the optimality and unfeasibility proofs obtained by the CP solver which confirm the efficiency of the MA, then show how the MA scales with larger and more complex instances.

A. Proved Optimal Solutions and Infeasibility

With an improved implementation of the arc-consistency algorithm (the main inference technique used with CP), our constraint program was able to obtain optimality proofs for all the instances of the 2013 benchmark and for problems up to 30 aircraft of the new 3D benchmark. However, within the 300 s allocated to run the algorithm, only the optimal solution could be obtained for most of these 3D instances, the proof of optimality generally being much longer. For greater number of aircraft (more than 40), even finding a first solution can be challenging and out of reach within the allocated time.

Figure 10 shows the mean time (in seconds) needed to compute the optimal solution for both the MA and CP algorithm for all instances ranging from 15 aircraft to 30 aircraft with uncertainty level 1. Whereas the MA scales well with the size of the instance, the performance of the constraint program quickly deteriorates. 40-aircraft instances are not shown in the figure, as only a few ones could be solved to optimality, the proofs taking several hours to one day to complete.

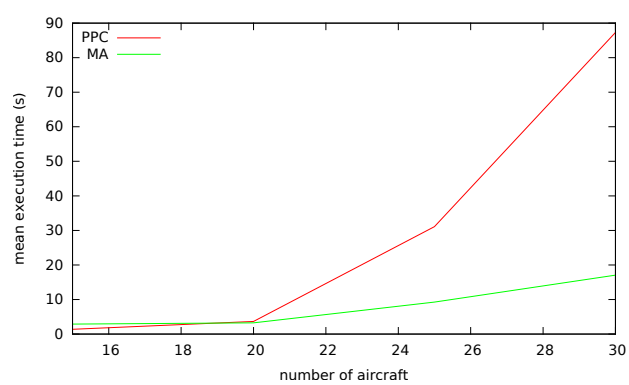

Fig. 10. Execution times to find optimal solution.

However, proofs were obtained in less than one hour by the CP algorithm for all unfeasible instances with 100 aircraft and uncertainty level 3 (with up to 8.8×10^6 forbidden trajectories pairs). These proofs validate the results of the MA solver,

which was able to obtain valid (and probably optimal) solutions on all feasible instances (though not within the allocated 300 s for 100-aircraft ones) and failed to do so on unfeasible problems.

B. Towards Large-Scale Problems

The Memetic Algorithm exhibits a much better behavior when the number of aircraft increases and was able to obtain conflict-free solutions for all feasible instances within 300 s, consistently achieving the optimum whenever the CP algorithm could prove it. Figure 11 shows the mean cost over all instances with increasing number of aircraft for the three uncertainty levels.

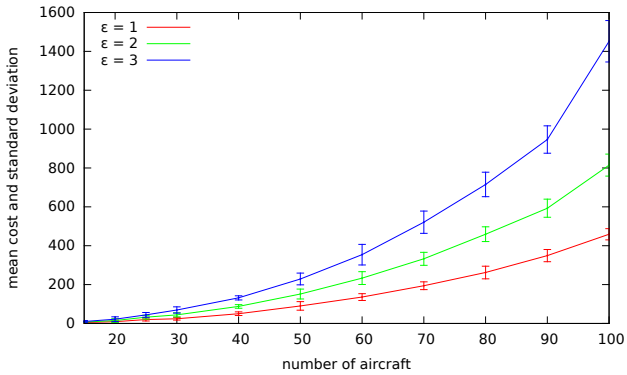

Fig. 11. Mean cost (and standard deviation) found by the Memetic Algorithm with 300 s time limit.

As expected, the cost of conflict resolution, which is the sum of the costs of all maneuvers, increases with the number of aircraft and with the uncertainty level. Moreover, as shown in figure 12 where the cost per aircraft is indicated, this growth is not constant because the density of trajectories increases while the airspace volume is kept constant, leading to more involved maneuvers.

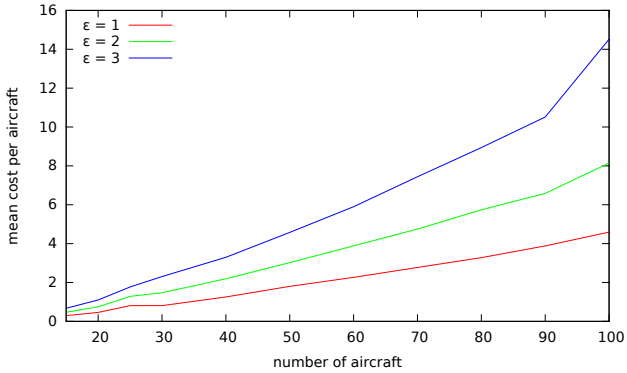

Fig. 12. Mean cost per aircraft found by the Memetic Algorithm with 300 s time limit.

Among the 5 runs executed on each instance, we observed that the MA consistently finds the same cost for problems with less than 50 aircraft, but the standard deviation amounts to almost 10 % for larger instances. Due to the large scale and number of forbidden trajectories pairs of the hardest problems, the local search performed by the MA can be trapped in local optima on some of the runs and probably fails to achieve optimality.

Moreover as shown on figure 13 for one of hardest instances, the Memetic Algorithm needs much more time than 300 s to converge to its best solution. The figure represents the convergence of the best solution encountered by MA during one run for instance #9 with 100 aircraft, uncertainty level 3 and a time limit equals to 2000 s. Note that if the execution time of the MA is limited to 300 s, then the best solution encountered so far is not a valid solution as one conflict remains and its cost is very high compared to the one found around 1100 s.

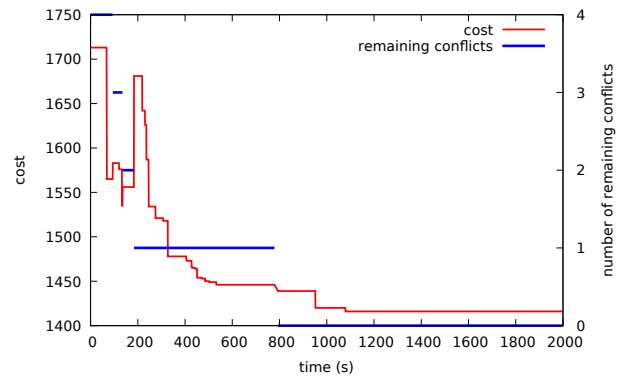

Fig. 13. Convergence of the Memetic Algorithm on a typical run for instance #9 with 100 aircraft and uncertainty level 3 and 2000 s time limit.

Furthermore, figure 13 shows how the MA is guided; firstly, the MA tries to decrease the number of remaining conflicts even if it increases the cost (e.g. at ≈100 s and ≈180 s); secondly, when the solution is admissible, the MA attempts to improve the cost value without reintroducing any conflict, exploring only the valid regions of the search space.

However, our results show that the MA solver performs well enough on reasonable instances corresponding to real-life scenarios and produces near-optimal solutions quickly enough for a real-time system.

VI. Conclusion and Further Work

We have presented a 3D extension of the en-route conflict resolution framework introduced in [9] with a more complete uncertainty model and the results obtain by our former Constraint Programming (CP) algorithm and a new metaheuristic, a Memetic Algorithm (MA), on the corresponding 3D instances.

On the one hand, CP was able to obtain optimality proofs within a time limit of 300 s for problems up to 30 aircraft of the new 3D benchmark. For larger instances, some optimality and unfeasibility proofs could be achieved if the solver was allowed to run for several hours, and others seem out of reach. On the other hand, the MA could always obtain solutions without conflict for all feasible instances, even with 100 aircraft, within 300 s, even consistently reaching the optimum whenever it could be proved by the CP solver.

The performance of such metaheuristics paves the way for the implementation of conflict solvers as a component of real-time systems, but they still need to be tested iteratively within a rolling horizon context on a full day of real traffic instead of artificial (though quite hard) unrelated scenarios.

References

[1] H. Erzberger, "Conflict probing and resolution in the presence of errors," in *Procceedings of the 1st USA/Europe Seminar*, 1997.

[2] N. Durand, J.-M. Alliot, and J. Noailles, "Automatic aircraft conflict resolution using genetic algorithms," in *Proceedings of the Symposium on Applied Computing, Philadelphia*, ACM, 1996.

[3] G. Granger, N. Durand, and J. Alliot, "Optimal resolution of en-route conflicts," in *4th ATM R&D Seminar*, 2001.

[4] L. Pallottino, E. Féron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 3–11, 2002.

[5] A. Vela, S. Solak, W. Singhose, and J. Clarke, "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, 2009.

[6] A. Alonso-Ayuso, L. Escudero, and F. Martin-Campo, "Collision avoidance in air traffic management: a mixed-integer linear optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 47–57, 2011.

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

[8] N. Durand and G. Granger, "A traffic complexity approach through cluster analysis," in *5th ATM R&D Seminar*, 2003.

[9] C. Allignol, N. Barnier, N. Durand, and J.-M. Alliot, "A new framework for solving en-routes conflicts," in *10th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2013.

[10] T. Lehouiller, J. Omer, F. Soumis, and G. Desaulniers, "A flexible framework for solving the air conflict detection and resolution problem using maximum clique in a graph," in *11th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2015.

[11] N. Barnier and C. Allignol, "Trajectory deconfliction with constraint programming," *The Knowledge Engineering Review*, vol. 27, no. 03, pp. 291–307, 2012.

[12] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," in *Information Processing Letters*, 1992.

[13] J.-K. Hao, *Memetic Algorithms in Discrete Optimization*, p. 7394. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[14] P. Van Hentenryck, "Constraint solving for combinatorial search problems: a tutorial," in *Principle and Practice of Constraint Programming CP'95* (U. Montanari and F. Rossi, eds.), vol. 976 of *Lecture Notes in Computer Science*, (Cassis, France), pp. 564–587, Springer, September 1995.

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

Authors

Cyril Allignol is an assistant professor at ENAC. He graduated from ENAC as an engineer in 2006, and received a PhD (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 PhD 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 is a professor at the ENAC/MAIAA lab. He 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 (then DSNA/DTI R&D) from 1992 to 2008, holds a PhD in Computer Science (1996) and got his HDR (French tenure) in 2004.

Alexandre Gondran is an assistant professor at ENAC. He graduated from TELECOM ParisTech as an engineer in 2001, and received a PhD (2008) in computer science from the University of Technology of Belfort-Montbliard and the University of Franche-Comté.

Ruixin Wang is a PhD student at École Nationale de l'Aviation Civile (ENAC) working on the cooperation between combinatorial optimization solvers. He graduated from ENAC and Civil Aviation University of China in 2016.