A ground holding model for aircraft deconfliction

Nicolas Durand, Cyril Allignol, Nicolas Barnier

A GROUND HOLDING MODEL FOR AIRCRAFT DECONFLICTION

Nicolas Durand, DSNA/DTI/R&D, Toulouse, France
Cyril Allignol, DSNA/DTI/R&D, Planning Optimization Modeling Team, Toulouse, France
Nicolas Barnier, Laboratoire d'Optimisation du Transport Aérien, ENAC, Toulouse, France

Abstract

In the SESAR traffic growth predictions, traffic complexity will become an issue that the current Air Traffic Management organization is not able to handle. The 4D trajectory concept offers new perspectives for deconflicting the traffic by ground-holding aircraft before they take-off. This paper studies the possible complexity reduction achievable by optimizing the aircraft take-off times. Therefore a simple model is introduced to detect pairwise 3D possible conflicts and define conflicting take-off time differences. Two resolution algorithms are tested on a real traffic data sample collected in the French airspace. The first one is based on a Constraint Programming model of the problem and ensures the optimality of the maximum delay required to solve every conflict. The second one uses an evolutionary computation algorithm to minimize the mean delay among the aircraft population. A sliding window model is introduced to reduce the size of the problem and to regularly update the current situation. Experimental results performed in the French airspace with fast time simulation show that with perfect 4D trajectory, every conflict over flight level 290 can be solved by delaying less than a quarter of the traffic within a range of delays varying from 1 to 90 minutes and a mean delay of 4 minutes. The Constraint Programming approach gives better results than the evolutionary computation approach. Adding uncertainty around 4D trajectories dramatically degrades the results.

Introduction

The SESAR program is anticipating the future growth of air traffic in Europe. In this context, the Episode 3 [1] project has aimed at assessing the performance of new ATM concepts, such as 4D trajectory planning and strategic deconfliction. ATC performances in Europe are limited by the hourly capacity constraints defined on each en-route ATC sector to limit the rate of entering aircraft. The current ground holding slot allocation process ensured by the CFMU (Central Flow Management Unit) allocates departure slots (that must be respected within a -5/+10 minutes margin) in order to respect these limits. Much research has been conducted to optimize the CFMU algorithm [2], [3]. However these approaches did not address the conflict pair resolution problem, but only try to respect sector capacity limits not necessarily related to the traffic complexity [4].

In the context of 4D trajectories, solving conflicts far in advance might become possible and prevents from defining complexity indicators that are hard to validate. Current equipment does not allow aircraft to fly perfect 4D trajectories and the results obtained in this arcticle can only give a theoretical cost of complexity reduction. The Air Traffic Management System will probably always need to be structured with different levels of granularity and time horizons in order to be able to handle uncertainties, unpredictable events and emergencies.

In the following sections, we briefly introduce the related work and the context of the research project. Then we give a description of the model chosen. Next we introduce the Evolutionary Computation approach and the Constraint Programming model that are compared using real data in the following section. The last section deals with sensitivity of the results to uncertainties. We conclude by outlining the further work that needs to be adressed to enhance the approach.

Related Work and Context of the Study

Many results were obtained using Constraint Programming (CP) approaches in the air traffic management domain. [3] has for example introduced a new approach to address the sector capacity constraint that prevents the peak effect due to the usual

definition of sector capacities (number of aircraft entering a sector per time period). CP approaches have also been successfully used to optimize the number of flight levels necessary to separate the routes having the biggest traffic flows [5]. Evolutionary algorithms have been successful in many different applications, such as short term conflict detection and resolution ([6], [7]), 3D conflict free network definition ([8], [9]) or taxiing optimization on big airports [10]. All these problems involve large data sets, are combinatorial, and deal with a large number of constraints that are difficult to handle with classical optimization methods.

In Europe, en-route sector capacities are limited below bounds defined over given time periods (usually one hour) by local experts, the Flow Management Position (FMP) Controllers in each Air Traffic Control Center (ATCC). The CFMU first identifies the overloaded sectors with the CFMU pre-tactical tool for network optimization PREDICT. The Computer Assisted Slot Allocation (CASA) tool [11] computes a slot allocation for some aircraft involved in the most constrained areas. CASA is able to deal with many operational constraints and updates, but is based on a greedy algorithm that cannot guarantee a solution that satisfies all the constraints or that is optimal.

The traffic complexity is difficult to measure and the sector capacities, expressed as a number of aircraft entering the sectors over a time period do not necessarily reflect the controller's workload. Gianazza [4] compared this measure to the moments when subsets of sectors are split or merged and showed very different complexity profiles. Recent work such as [12] uses a more precise and complex CP model to balance the traffic over the sectors in a control center of the upper airspace. Barnier [13] used CP to optimize the sectors opening schedules to match more closely the predicted traffic. Baptiste and Tan Dac [14] also used CP to redesign airspace sectorization with a better balancing.

In this paper, we propose to estimate the cost of directly solving all conflicts in the upper airspace with ground holding, provided that aircraft are able to follow their trajectories accurately. Therefore we introduce a model of this large scale combinatorial optimization problem based on a sliding window principle.

Modeling the problem

In the model, aircraft estimate their 4D trajectories a few minutes before departure and send them to a centralized unit that first detects 3D pairwise potential conflicts and check if the aircraft take-off times are compatible. In many situations, no 3D conflicts occur, or the current take-off times of the pair of aircraft are not conflicting. When a 3D-conflict occurs, the difference of take-off times creates a constraint between the aircraft pair concerned. Clusters of conflicts are the transitive closure of the potential 3D-conflict pairs.

A. Conflict Detection

Real traffic data are provided by the CATS[1] simulator [15], which takes as input all of the flight plans in a given airspace for a given day of traffic, and outputs the corresponding 4D trajectories.

To detect all "significant" conflicts, CATS discretizes trajectories with a $\delta t = 15\,s$ time step, which is small enough to detect every conflict.

Trajectories are then probed pairwise[2] for potential conflicts, ignoring those that could only occur for greater delays than the given maximal one Δ_{max}. The separation norm is thus tested for each pair of points of the two probed trajectories (up to $p = 1300$ points per trajectory for up to $n = 9500$ flights in $\mathcal{O}(n^2p^2)$, as observed in the largest instances simulated by CATS) as illustrated on figure 1 in the horizontal plane. Note that trajectory enclosing bounding boxes or sweep line techniques [16] could be used to lower the detection complexity, but it is here considered as a static data production phase and its efficiency is not a primary concern of the present study.

Though the maximal allowed delay Δ_{max} can be seen as a parameter of the search algorithm only, it also affects conflict detection. Indeed, when the maximal allowed delay is increased, the size of the problem grows as well, as more and more flights tend to be in potential conflict. Ultimately, if a 24 h-delay would be allowed, the conflict detection could be done regardless of time, as any two space-conflicting trajectories would generate a constraint.

[1]The Complete Air Traffic Simulator developed at DSNA/DTI.

[2]Note that the conflict detection for two given flights is symmetrical, so that only ordered pairs are considered.

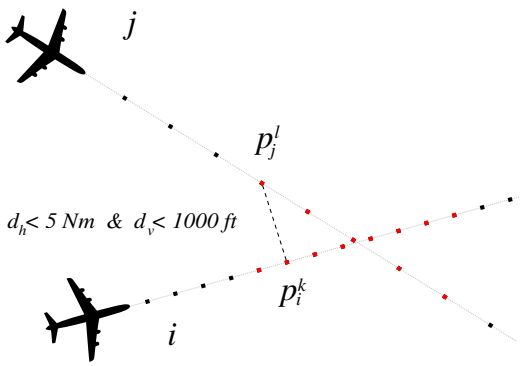

Figure 1. Detection of conflicting points.

So, whenever a particular instance has been proved inconsistent, it has to be generated again with higher values of the maximal delay, which will capture later potential conflicts on the trajectory pairs and increase the size of the instance.

Operationally, flights originating outside the Eurocontrol countries cannot be delayed, so their delay variable will be fixed to 0 in our constraint model, reducing the number of variables but tightening the constraints as well and offering less opportunities for optimization. Constraints corresponding to conflicts occurring between two such flights will of course be discarded as we cannot delay the flights to solve them. Such remaining conflicting cases would have to be taken care of by other ATC or ATFM techniques that will not be addressed in this study.

B. Conflicts Constraints

To compute the constraints of our model, the trajectories (up to 9500 flights for one day of traffic in the French airspace) are pairwise probed for couples of conflicting points.

We define a set D of decision variables:

$$D = \{\delta_i, \forall i \in [1, n]\}$$

of finite domain $[0, max\_delay]$ that represents the delay associated with each of the n flights (between 6000 and 8000 after processing for our experiments). The value of max_delay is typically chosen as 90 min and increased as needed when no solution is found (up to 300 min in our experiments). As explained in the previous section, the size of the instance grows with max_delay.

It was shown in [17] that pairwise conflict constraint could be expressed in the following way: if δ_i and δ_j are respectively the delays of aircraft i and j then aircraft i and j are not in conflict if

$$d_{ij} = \delta_i - \delta_j \notin [\underline{lb}^1, \overline{ub}^1] \cup \ldots \cup [\underline{lb}^q, \overline{ub}^q]$$

where $[\underline{lb}^k, \overline{ub}^k]$ are time intervals representing different possible conflict zones between the two aircraft. Most of the time, $q = 0$, which means that the two trajectories are not conflicting or $q = 1$ which means that they have one conflicting zone. When a pair of flights are conflicting on several disjoint times over their entire trajectories (as illustrated on figure 2), $q > 1$.

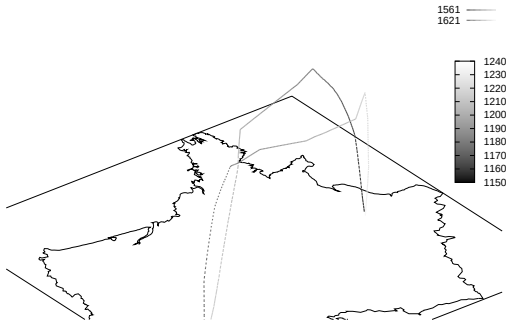

Figure 2. Three potential conflicts between two flights: one near Paris airport at low altitude and two other en-route at the cruising altitude of the lower flight. The gray scale corresponds to time (in minutes) along the trajectory, the lighter the later.

1) Further Instance Processing: The takeoff and landing part of trajectories are truncated around airports within a given radius (usually 10 NM) as the traffic is considered to be handled with specific procedures by the TMA control services in these zones.

After the computation of the conflict constraints, the whole instance is scaled down to a more reasonable time step than the 15 s used during conflict detection. We ensure that the original forbidden intervals are strictly included in the scaled ones, with appropriate interval arithmetic operations which can possibly merge conflicting disjoint intervals. A 1 min time step was chosen for the resolution: a smaller time step would not have been realistic for a departure time, whereas a greater one would have reduced the set of feasible solutions. Furthermore,

a 1 min time step seems compliant with the SESAR objectives of ± 3 min precision for takeoff time.

In addition, the flight level of the detected conflicts were filtered, for example to only take into account conflicts occurring within the upper airspace (from FL290 and above). The minimal and maximal altitude of each conflict is recorded during the detection stage and a conflict is discarded if it entirely occurs below or above the specified airspace slice.

It is also possible to filter the time interval during which the conflicts may occur, taking the time bounds of the allowable delay into account. Any conflict strictly occurring outside the given time interval is then discarded.

Eventually, all the flights that do not have any conflict with any other flight are withdrawn from the instance.

2) Conflict Extension: To improve the robustness of our solutions towards uncertainty on the departure times of the flights, we add an extra parameter ext that extends conflicting intervals by a fix amount of time. Such an extension of ext minutes stretching the start and the end of a conflict will be able to manage uncertainties of $\pm\frac{ext}{2}$ minutes on the departure slot (see the last section), at the price of an increase in the cost of the solutions.

We expect that this extension scheme may as well be able to diminish the effects of other sources of uncertainty (e.g. vertical and ground speed) regarding the number of remaining conflicts during simulation.

C. A Sliding Forecast Time Window model

In order to limit the size of the problem and to be reactive to uncertainties, the whole day is not treated at once, but only aircraft scheduled for take-off during the next T_w minutes are considered. T_w represents the lookahead time also called *forecast time window*. The situtation is reconsidered every σ minutes with $\sigma << T_w$. σ is the time step used in the model to make the T_w time window *slide*. This approach ensures that the problem can be updated every σ minutes : if an aircaft needs to cancel or delay its departure for external reasons, it will be able to free its slot and be reconsidered later on. Aircraft that are already airborne at the current time are taken into account as constraints. This is also the case for aircraft scheduled in the next A_n minutes (see figure 3) because an advance notice time is required to assign any delay to an aircraft. In practice, this advance notice time A_n can be longer or shorter than σ. In the numerical results, this value was set to 0 because it does not affect the quality of the solution. Figure 3 gives an example of the evolution of an aircraft take-off slot when the forecast time window slides. At the first step (current time=0) the aircraft is delayed to take-off at T_1. Because T_1 is far enough from the current time (more than $A_n + \sigma$ minutes ahead, in the dotted zone), the slot will still be modifiable when the current time is σ (second line). When the time window slides (current time=σ), the delay might be reduced and a slot chosen at T_2 which is still modifiable. At current time $2\,\sigma$ the delay can be reduced again and a slot chosen at T_3 which belongs to the notification zone. After this current time (2σ) no modification is possible because at the next step (current time=3σ), the take-off slot will be too close to the current time (less than the advance notice required).

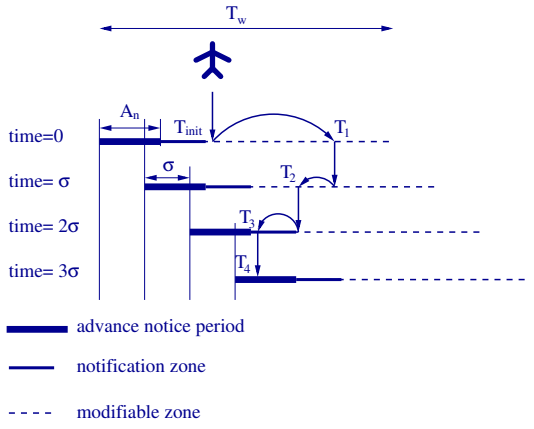

Figure 3. Sliding forecast time window.

The size of the forecast time window is an important parameter. If it is too big, the size of the problem will include a very large number of variables and the resolution might be more difficult. If it is too small, the solutions found might be worse and the total delay induced over the day much higher. This will be debated in the results section.

Evolutionary Computation Approach

Classical Evolutionary Computation principles such as described in the literature [18], [19] were

used for this approach.

D. Fitness function

The cost function used in this part is simply the sum of the delays over the aircraft population.

Solutions respecting the separation constraints cannot be built easily. Consequently, we need to include the separation criteria in the fitness function.

The fitness function chosen is:

$$F = \frac{n - \sum_{i=1}^{n} (\frac{\delta_i}{\delta_{max}})}{1 + nrc}$$

where n is the number of aircraft and nrc is the number of remaining conflicts.

The fitness function increases when the number of remaining conflicts and delays decrease. It takes its values in $[0, n]$.

E. Crossover operator

The conflict resolution problem is partially separable as defined in [20], [21]. In order to increase the probability of producing children with a better fitness than their parents, principles applied in [20] were used. For each aircraft i of a population element, a local fitness F_i value is defined as follows:

$$F_i = \frac{1 - (\frac{\delta_i}{\delta_{max}})}{1 + nrc_i}$$

where nrc_i is the number of remaining conflicts involving aircraft i.

Figure 4 presents the crossover operator. First two population elements are randomly chosen. For each parent A and B, fitness A_i and B_i of aircraft i are compared. If $A_i < B_i$, the children will take aircraft i of parent A. If $B_i < A_i$, the children will take aircraft i of parent B. If $A_i = B_i$ children randomly choose aircraft A_i or B_i or even a combination of A_i and B_i.

F. Mutation operator

For each candidate to mutation, the delay of an aircraft having one of the worst local fitnesses is modified. If every conflict is solved, an aircraft is randomly chosen and its parameters changed. In practice, a number m is randomly chosen in the interval $[1, \frac{n}{2}]$ and we pick up m times an aircraft to find the most constrained aircraft among these m trials. The delay of this aircraft is then either locally optimized or randomly modified with a probability

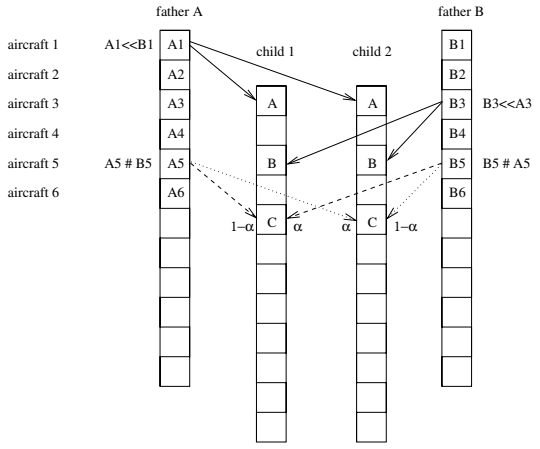

Figure 4. Crossover operator

of 50%. We could be tempted to always locally optimize the delay of the worst aircraft, but this would make the algorithm become very deterministic and lead to a premature convergence of the algorithm.

The crossover and mutation operators are more deterministic during the first generations because there are many conflicts to solve. They focus on making feasible solutions. When the solutions without conflicts appear in the population, they become less deterministic.

Sharing: The problem is highly combinatorial and may have many local optima. In order to prevent the algorithm from a premature convergence, the sharing process introduced by Yin and Germay [22] is used. The complexity of this sharing process has the great advantage to be in $n \log(n)$ (instead of n^2 for classical sharing) if n is the size of the population. The distance used to compare two population elements p and q is:

$$D = \frac{\sum_{i=1}^{n} |\delta_i^p - \delta_i^q|}{n}$$

G. Parameters

In the experiments, the following parameters were empirically chosen: the size of the population was set to 100, 20% of the population is crossed, 60% is muted, the selection uses the *stochastic remainder without replacement*. A sharing process is used. As time to solve a problem is limited, the number of generations is limited to 500.

Constraint Programming strategy

H. Cost Function

For this approach, the cost is simply defined as the maximal allocated delay, to ensure equity among the various postponed flights:

$$cost = \max\{\delta_i, \forall i \in [1, n]\}$$

However, the overall sum of the delays is of utmost importance as well for the quality of a solution and we will take it into account within the search strategy to provide realistic max-optimal solutions during the resolution of the problem.

I. Search Strategy

The constraints of the problem are reminiscent of the disjunctive mutual exclusion constraints often used to model scheduling problems [23]. At a coarse grain, we could consider each conflicting area as a machine on which to process two tasks of different lengths (depending on the speed of the aircraft). Several conflicts along a trajectory could even be seen as the ordered tasks of a given job, as in the Job-shop Scheduling Problem (JSP).

However, the comparison does not hold much further. First, the time intervals between any two conflict-tasks of the same trajectory are fixed, as only one delay variable is associated with each flight (unlike the JSP where all tasks are only related with precedence constraints). Second, to consider a potential conflict in three dimensions only, as the transitive closure of the overlapping conflicting segments, with task lengths proportional to the time spent by the aircraft within the area, is misleading. In this setting, the conflict associated with two catching-up flights on the same route would be the entire trajectory, preventing them from being airborne at the same time! Obviously, our model is much more precise and allows two aircraft on the same route to be separated by 5 NM only. Third, the number of "conflict machines", if not quadratic in the number of "flight jobs" as it could ultimately grow for arbitrary instances, is quite large and cannot be easily related with any known standard scheduling problem.

Nevertheless, the branching scheme of our search strategy to solve this essentially disjunctive problem is inspired by standard scheduling techniques. For instance, trying to start the search by directly labelling the delay variables δ_i may impede the search because of thrashing (i.e. repeatedly fail over the same constraint), as the constraints are expressed over the differences d_{ij}. In this case, a more efficient filtering can be obtained by feeding the propagation of the arithmetic constraints with new domain bounds for the d_{ij} auxiliary variables.

In this respect, our search strategy first tries to order pairs of conflicting flights by adding the constraint $d_{ij} < \underline{lb}$ or $d_{ij} > \overline{ub}$ in the case of a single conflicting interval. If there are several holes in the domain of d_{ij}, branching is repeated with the bounds of the remaining holes. The variable d_{ij} with the highest sparsity, i.e. the smallest ratio between the domain size and the difference of the domain bounds, is chosen first for branching.

To compensate for the cost being defined as the maximal delay only, disregarding the total amount of time, we choose to branch first within the d_{ij} interval corresponding to the minimum potential increase for its delay variables δ_i and δ_j. Such an interval would be the closest to 0, while if d_{ij} were far from 0, then at least one delay would be large. Whenever the search backtracks over such a decision, this interval is discarded and we branch on the next one recursively.

When all conflicts are ordered and there are no more holes in the domain of the d_{ij}, we start labelling the decision variables δ_i with a standard dom/deg selection heuristic: the variable with the smallest domain, and the highest number of constraints in case of tie, is dynamically selected. Then, the values closest to 0 are probed first to attempt to keep the total amount of delay as low as possible.

After the first solution is found, the branch and bound algorithm proceeds by dichotomy on the cost domain to find the optimal solution with respect to minimization of the maximal allocated delay, while keeping low the overall amount of delay thanks to the search strategy.

Experimental results

J. Evolution of the problem size

Simulations were performed using real flight plan data on the Complete Air Traffic Simulator (CATS) developed by the French DSNA. Simulations were

run on the French airspace on a heavy day of traffic (October 10, 2008). Figure 5 shows the evolution of the number of aircraft that can be delayed during the day, with different forecast time windows. The problem sizes increase until 5 a.m. and reach a peak at 8 am. They decrease at the end of the day when the traffic starts decreasing. 8 different time windows were tested from 15 to 360 minutes and also considering the whole day at once (1440 minutes). Figure 6 shows that the number of aircraft that can be delayed (dotted line) and the total number of aircraft (solid line) at the peak hour (8 am) grow linearly with the forecast time window width (in minutes).

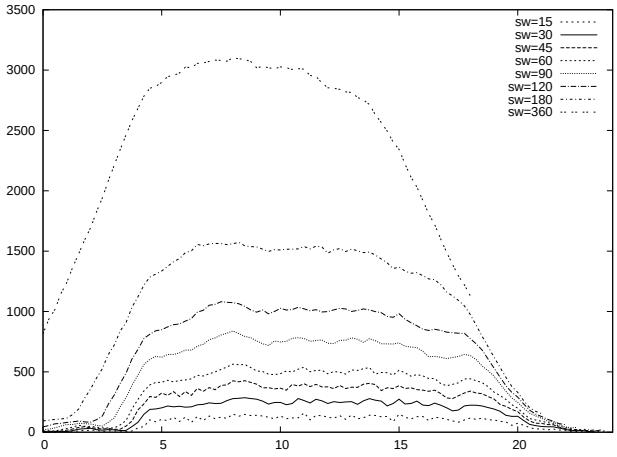

Figure 5. Evolution of the problem size for different forecast time windows.

Simulation results also show that it is not possible to separate the traffic into smaller problems that can be solved independently because the 3D-traffic clusters generally involve most of the current traffic, especially during peak hours.

K. Comparison of results with no uncertainty

Table I gives the number of aircraft delayed, the maximum delay, the total sum of the delays, the mean delay over the delayed aircraft, and the computation time (on an Intel Xeon 2.66GHz processor) for different time windows. In the simulation, only conflicts over $FL290$ were considered, and $\sigma = 15$ min. The problem had 6184 variables over the day for 8693 flights.

With the CP approach, all the conflicts can be solved by delaying less than a quarter of the aircraft.

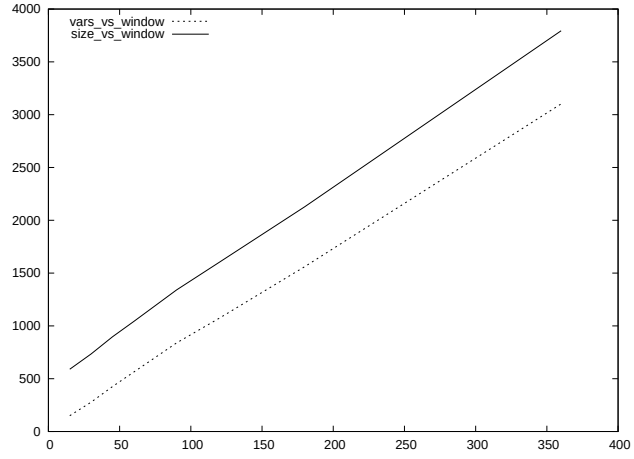

Figure 6. Total number of aircraft (solid) and problem size (dotted) at 8 am as a function of the forecast time window size.

T_w (min)	nb acft delayed	Max delay	Sum (min)	Mean/ delayed	Comput time(s)
CP					
90	1392 (22%)	88	5102	3.67	2096
120	1390 (22%)	84	4782	3.44	2435
180	1391 (23%)	84	4739	3.41	2830
360	1388 (22%)	84	4785	3.45	6093
1440	1385 (22%)	84	4677	3.38	24
EC					
30	1184 (19%)	87	8197	6.92	21584
45	1153 (19%)	84	9205	7.98	32264
60	1119 (18%)	90	10146	9.07	43123
90	1091 (18%)	88	10673	9.78	69853
120	1101 (18%)	88	11755	10.68	97153
180	1097 (18%)	88	13731	12.52	172675
360	1092 (18%)	88	17103	15.66	470355
1440	1076 (17%)	89	35590	33.08	507155

TABLE I
COMPARISON OF RESULTS FOR THE 2 APPROACHES AND DIFFERENT T_w

The mean delay is smaller than 4 minutes and the maximum delay smaller than 88 minutes. When increasing the Forecast Time Window T_w the mean delay decreases but the computation time increases.

With the EC approach, all the conflicts can be solved by delaying less than a fifth of the aircraft. The mean delay is close to 7 minutes for $T_w = 30$ min but it increases with T_w. An explanation could be that when the size of the problem increases the EC algorithm does not converge as easily to a good solution and returns a solution that is worse

Ext	Delayed	Max del	Sum (min)	Mean/ delayed	Time (sec)	Rem Conf
CP - 90 minutes sliding window						
2	3543-57%	244	45789	12.92	3181	0
4	4581-73%	351	286912	62.63	5364	0
CP - whole day optimization						
2	3514-56%	88	42634	12.13	3499	0
4	4526-72%	336	283907	62.73	2294	0
6	4866-78%	817	759753	156.14	2676	0
EC - 90 minutes sliding window						
2	2437-39%	90	81405	33.40	111401	0
4	2543-41%	90	90463	35.57	119171	279
6	2497-40%	90	89181	35.72	124746	592
EC - whole day optimization						
2	1355-22%	500	371669	274.29	87012	921
4	1056-17%	500	287232	272.00	99077	3573

TABLE II
COMPARISON OF APPROACHES FOR DIFFERENT UNCERTAINTY PARAMETERS, WITH A 90 MINUTES SLIDING TIME WINDOW OR FOR THE WHOLE DAY (1440 MINUTES).

than when the size of the problem is smaller. The computation times are much higher than those obtained with the CP approach.

Results show that the CP approach is much more efficient in time resolution, total and mean delay, but tends to delay more aircraft than the Evolutionary Computation approach. The maximum delay is slightly higher in the EC approach. The resolution time increases with T_w in both approaches but the growth is higher for the EC approach.

L. Influence of uncertainty

Uncertainties on take-off times are very frequent and can be caused by many different factors. In order to make the previous approach robust to uncertainties, we need to prevent aircraft from being in conflict even if their take-off times are not precise. Therefore the value of the ext parameter (described in the modeling section) in the model should be increased.

Table II compares results for different values of ext with a 90 minutes sliding time window or for the whole day (1440 minutes) of traffic.

An uncertainty of ± 1 minute ($ext = 2$) requires to delay more than twice as many aircraft as with no uncertainties, with the CP approach. With the Evolutionary Computation approach, the number of aircraft delayed is almost twice the number found with no uncertainties. The delay increase is huge (9 times for the CP, and 8 times for the EC). For $ext = 4$ (uncertainty of ± 2 minutes on the take-off time), delays obtained with the CP approach are far too big to be acceptable in a real context. The EC algorithm cannot even solve every conflict within a reasonable computation time.

The target figures for efficiency within SESAR are:

- at least 98% of flights departing on time (on-time departure being defined as actual departure less than 3 min before or after scheduled departure),
- the average departure delay of delayed flights must not exceed 10 min.

The first target would correspond to $ext = 6$ minutes. Our results show the limits of the concept. In order to meet this target, we probably need to combine our strategy to others such as, for example, an efficient flight level allocation in order to separate the main flows of aircraft.

Conclusion

This article introduces an original ground holding approach to solve all potential conflicts occurring above a given flight level a few hours or minutes in advance with a sliding window model. Rather than trying to respect sectors capacities constraints, we model each possibly conflicting situation between any two aircraft and impose adjustements of departure times to keep them separated, with the hypothesis that aircraft could precisely follow their planned 4D trajectories. The resulting problem size can be reduced by only looking at the next T_w minutes and reconsidering the situation every σ minutes.

Two different algorithms were compared to allocate the delays to aircraft. The first one is an evolutionary algorithm that uses the partial separability of the problem to define efficient crossover and mutation operators. The other one is based on a Constraint Programming (CP) approach and uses the FaCiLe constraint library developed at ENAC. It is based on a branch and bound algorithm that minimizes the maximum delay on every aircraft.

Simulations were performed using real flight plan data on the Complete Air Traffic Simulator (CATS)

developed by French DSNA. Simulations were run on the French airspace on a heavy day of traffic (October 10, 2008) focusing on conflicts over Flight Level 290. The sample dealt with 8693 flights that generated 1418 conflicts above FL 290. Results using the Constraint Programming approach are better than those obtained with the Evolutionary Algorithm. They show that every conflict could be solved by delaying 22% of the flights with a mean delay of 4 minutes and a maximum delay of 84 minutes.

We have also presented a first step toward taking uncertainties into account by extending the forbidden intervals of conflicting flights. However, an extension as small as 4 min, which is able to cope only with a 2 min uncertainty on the departure time generates tremendous amounts of delays, far above SESAR performance objectives. We plan to overcome these issues and further assess the possible outcomes of 4D trajectory planning in the context of Episode 3 WP4 and address larger (European) instances with various techniques like combining our delay algorithm with a prior flight level allocation, repeatedly solving the problem on a sliding time window or solving the remaining conflicts with a CATS resolution module.

This research is a primary approach to reduce the traffic complexity by solving conflicts with ground holding. Simulations on real data showed that most conflicts could be solved by ground holding aircraft in a 4D trajectory context. However the model remains very unrealistic since it does not take into account many uncertainty parameters. For example winds can largely modify the trajectory predictions. Further work is currently being done on the robustness of such solutions, that can be achieved by reducing the update frequency of the model and increasing the size of forbidden take-off time intervals.

References

[1] R. Graham and D. Young, "Preparing an initial assessment of the SESAR concept of operations "EP3: Single european sky implementation support through validation"," Eurocontrol Experimental Centre, France, Tech. Rep., 2006.

[2] M. Dalichampt, E. Petit, U. Junker, and J. Lebreton, "Innovative slot allocation (ISA)," Eurocontrol, Tech. Rep., 1997.

[3] N. Barnier, P. Brisset, and T. Rivière, "Slot allocation with constraint programming: Models and results," in *International Air Traffic Management R&D Seminar ATM-2001*, Santa Fe (NM), USA, December 2001.

[4] D. Gianazza and K. Guittet, "Selection and evaluation of air traffic complexity metrics," in *25th DASC*, 2007.

[5] N. Barnier and P. Brisset, "Graph coloring for air traffic flow management," in *CPAIOR'02: Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems*, Le Croisic, France, March 2002, pp. 133–147.

[6] N. Durand and J. Alliot, "Optimal resolution of en route conflicts," in *1rst USA/Europe Seminar*, 1997.

[7] S. Mondoloni and S. Conway, "An airborne conflict resolution approach using a genetic algorithm," in *AIAA Guidance, Navigation, and Control Conference*, August 2001.

[8] D. Gianazza and N. Durand, "Separating air traffic flows by allocating 3d-trajectories." in *23d DASC*, 2004.

[9] T. Riviere, "Redesign of the european route network for sector-less," in *23rd DASC*, 2004.

[10] J.-B. Gotteland and N. Durand, "Genetic algorithms applied to airport ground traffic optimization," in *CEC2003*, 2003.

[11] *Basic CFMU Handbook - General & CFMU Systems*, 6th ed., Eurocontrol CFMU, Brussels, February 2000.

[12] P. Flener, J. Pearson, M. Ågren, C. Garcia Avello, M. Çelitkin, and S. Dissing, "Air-traffic complexity resolution in multi-sector planning," *Journal of Air Transport Management*, vol. 13, no. 6, pp. 323–328, November 2007.

[13] N. Barnier, "Application de la programmation par contraintes à des problèmes de gestion du trafic aérien," Ph.D. dissertation, Institut National Polytechnique de Toulouse, December 2002.

[14] H. Tran Dac and P. Baptiste, "Airspace sectorization by constraint programming," in *RIVF'03*, 2003.

[15] J.-M. Alliot, J.-F. Bosc, N. Durand, and L. Maugis, "CATS: A Complete Air Traffic Simulator," in *16th DASC*, 1997.

[16] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry – Algorithms and Applications*. Springer, 1998.

[17] N. Barnier and C. Allignol, "4d-trajectory deconfliction through departure time adjustment," in *8th USA/Europe Air Traffic Management Research and Development Seminar*, 2009.

[18] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading MA Addison Wesley, 1989.

[19] Z. Michalewicz, *Genetic algorithms + Data Structures = Evolution Programs*. Springer-verlag, 1992.

[20] N. Durand and J.-M. Alliot, "Genetic crossover operator for partially separable functions," in *Genetic Programming*, 1998.

[21] 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.

[22] X. Yin and N. Germay, "A fast genetic algorithm with

sharing scheme using cluster analysis methods in multimodal function optimization," in *Proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference, Innsbruck Austria*, C. R. R.F.Albrecht and N. Steele, Eds. Springer-Verlag, 1993.

[23] P. Baptiste, C. Le Pape, and W. Nuijten, *Constraint-Based Scheduling, Applying Constraint Programming to Scheduling Problems*, ser. Kluwer's International Series in Operations Research & Management Science. Springer, 2001.