Modeling the Controller's Conflict Detection Task Using Fast Time Simulation

Nicolas Durand, Géraud Granger

MODELING THE CONTROLLER'S CONFLICT DETECTION TASK USING FAST TIME SIMULATION

Nicolas Durand, DSNA/DTI/R&D, Toulouse, France
Géraud Granger, Stéria, Toulouse, France

Abstract

This paper shows how the CATS (Complete Air Traffic Simulator) can model the controller's conflict detection task using a sliding time window. Because of uncertainties, detection is by far the most time consuming and cognitively challenging aspect of air traffic control. The processes governing detection are distinct from resolution actions. Few conflicts detected lead to a resolution maneuver. Because future aircraft positions are uncertain, controllers detect many more conflicts or "potential conflicts" than actually occur. In this paper, we solve conflicts in a real time context and compare different uncertainty scenarios. We show how uncertainties impact the number of theoretical potential conflicts detected and the increase the number of unnecessary maneuver actions required to keep the traffic safe. When we deal with realistic uncertainty values, the majority of calculation time in CATS is used for calculating maneuvers that will usually not be transmitted to pilots. This observation reveals the true nature of the controller's workload today.

Introduction

Much current research models and estimates the controller's workload. Some studies analyse questionnaires filled by the controllers themselves in different situations. Others use talking time measurements between controllers and pilots. In experiments, controllers have been fitted out with sensors to measure their heartbeat or eye movements [1], [2]. Existing models often divide the controller's workload into three specific tasks. The coordination task is clearly identified as the action of taking into account new aircraft, establishing contact with the pilot, and delivering the aircraft to the next sector with respect to existing rules. The monitoring task consists in making sure that the aircraft respects the trajectory defined by the flight plan. Modeling the monitoring task is not easy because an objective of trajectory monitoring is conflict detection. The detection-resolution task is the only task in which the controller can have a concrete effect on aircraft trajectory. Most people include conflict detection as part of the controller's conflict resolution task. However, this paper argues that conflict detection must be conceptualized as a specific task. Because of uncertainties, detection is by far the most time consuming and cognitively challenging aspect of air traffic control. The processes governing detection are distinct from resolution actions. Few conflicts detected lead to a resolution maneuver. Because aircraft future positions are uncertain, controllers detect many more conflicts or "potential conflicts" than actually occur. They generally try to avoid unnecessary maneuvers but they sometimes solve "fake" conflicts for two reasons: 1) usually maneuvers must be initiated far in advance of the conflicting zone. By the time a maneuver is decided, the controller cannot estimate precisely the aircraft's future positions; 2) because controllers are dealing with many aircraft at the same time, they often need to decide maneuvers that will definitely solve potential conflicts in order to give them more time to concentrate on other parts of the traffic. The controllers' priority is security rather than efficiency, they must use learned methods and experience and usually do not show initiative when solving a conflict.

In this paper, we show how the CATS fast time simulator models both the conflict detection and resolution task. By using a sliding window over a day of traffic, we can simulate the controller's behaviour in a real time context and compare different uncertainty scenarios. We show how uncertainties impact the number of theoretical potential conflicts detected and the increase of unnecessary maneuver actions required to keep the traffic safe. We used the French upper airspace to compare different scenar-

ios on different days of traffic. The simulations show many situations where maneuvers are anticipated and finally not executed because trajectory prediction is updated and potential conflicts disappear as the conflict zone approaches. When we deal with realistic uncertainty values, the most calculation time in CATS is used for calculating maneuvers that will usually not be transmitted to pilots. This observation reveals the true nature of controller's workload today. Many complexity studies have shown that the number of aircraft in a sector explains most of the controller's workload [3]. The monitoring task is thus very greedy compared to the resolution task itself because uncertainties create many potential conflicts and increase the monitoring and detection work of controllers. Today's new communication, navigation and positioning systems can help predict aircraft future positions accurately. New tools are tested and installed in countries around the world to assist controllers who are still responsible for conflict detection. In order to be effective and gain the trust of controllers, these tools should never fail to detect a conflict and must be better than or as good as controllers. The challenging question is not how close to trajectory prediction should aircraft stick in order to build an efficient detection resolution tool, but how accurate can trajectory prediction be to build tools that could help controllers in their detection task. The different stakeholders of the system sometimes have the feeling that Air Navigation System Providers want to constrain the trajectories. This paper shows the importance of better sharing uncertainty information.

In the first section we describe the general framework of the CATS simulator. The second part details the ERCOS solver that is used to build conflict free trajectories. The third part focuses on the solver algorithm. Part four details the experimental results performed with different uncertainty scenarios. We show how the controller's detection task can be modeled and how future TP improvements could decrease the controller's workload in the future.

I. The Complete Air Traffic Simulator (CATS)

A. General framework

The CAT Simulator was used in many projected conducted by the French DSNA to study either centralized control models [4], [5], [6] or Free-Flight concepts [7], [8], [9].

In this paper, we used real data traffic simulation. The CAT Simulator takes as input flight plans given by airlines or pilots before or after regulation. The simulator uses the BADA (Base of Aircraft DAta) tabulated model for modeling aircraft performances. For an aircraft type, it gives a vertical speed and a ground speed depending on whether the aircraft is climbing, leveled or descending. The BADA performance summary tables are derived from the total energy model of EUROCONTROL.

Aircraft speeds can be modified by a random value to take into account different factors of uncertainty (aircraft load, winds, etc. . .). These values can be either computed once at aircraft activation and remain the same for all the flight, or can be modified anytime during the flight. Uncertainty modeling for conflict detection and resolution is introduced in the next section. We will see how using uncertainty on trajectory prediction can model the controller's conflict detection-resolution task.

Aircraft follow classical routes (from way-point to way-point). The flight model is simple. An aircraft first climbs up to its RFL (Requested Flight Level), then remains leveled until its top of descent, then descends to its destination.

Aircraft flight is calculated per time step. The time step is always chosen in order to guarantee that the simumlator cannot miss any conflict. For all our simulations, we use a $15s$ time step.

B. General architecture of the system

We sketch here the general architecture of the simulator. Each part will be detailed in the following sections. The system architecture is presented in figure 1 and 2. The system relies on three main processes P1, P2, and P3:

- P1 is the traffic simulator.
- P2 is in charge of conflict pair detection, clustering of pairs, and verification of new trajectories built by the solver.

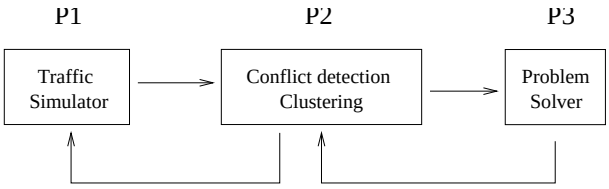


Figure 1. General architecture

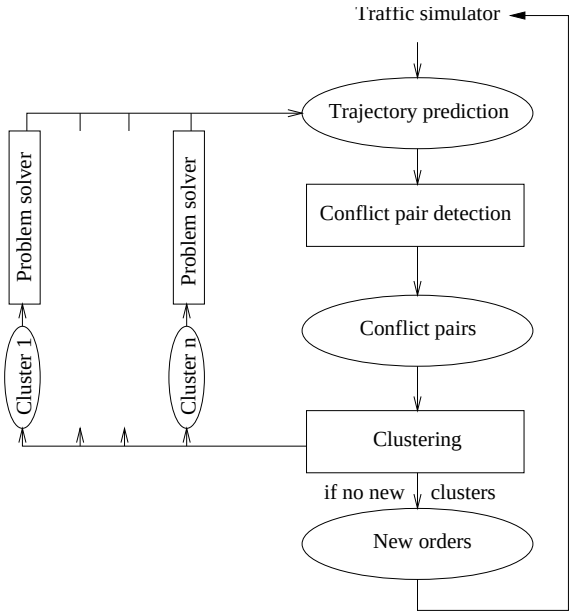


Figure 2. Detailed architecture of the prototype

- P3 is the problem solver.

P1 sends current aircraft positions and flight plans to process P2. Process P2 builds trajectory forecasts for T_w minutes, does conflict detection by pairs and transforms 1-to-1 conflicts in n-aircraft conflicts. Then, process P3 (the problem solver) solves in parallel each cluster, as aircraft in each cluster are independent from aircraft in the other clusters. The problem solver sends to P2 new orders and P2 builds new trajectory forecasts based on these orders. Then P2 once again runs a conflict detection process to check that modified aircraft trajectories do not interfere with aircraft in another cluster, or with new aircraft. If no interference is found, new flight orders can be sent to P1. If there are interferences, interfering clusters are joined and the solver is used again on that (these) cluster(s). The process is iterated until no interference between clusters remains, or no new aircraft is concerned by modified trajectories. The new orders are sent back to the traffic simulator.

The above process is iterated and all trajectories are optimized each δ minutes (3 or 5 minutes in the experiments). However, during the computation time, aircraft are flying and need to know if they must change their route or not. δ should be large enough to compute a solution, send it to the pilot and leave him enough time to begin the maneuver. Consequently, for each aircraft, at the beginning of the current optimization, trajectories are determined by the previous run of the problem solver and cannot be changed for the next δ minutes.

C. Conflict detection and clustering

1) Trajectory forecast and 1-to-1 conflict detection: As described above, the P2 process does trajectory prediction for T_w minutes. This trajectory prediction is done again by a simulation on a slightly modified version of the Air Traffic simulator. But, as stated above, we assume that there is an error about the aircraft's future location because of ground speed prediction uncertainties[1]. Climbing and descending rate uncertainties are larger than ground speed uncertainties. Because the conflict free trajectory must be robust regarding these and many other uncertainties, an aircraft is represented by a point at the initial time. The point becomes a line segment in the direction of uncertainty (the speed direction here, see figure 3). The first point of the line "flies" at the maximum possible speed, and the last point at the minimum possible speed.

When changing direction on a beacon, the heading of the line segment's "fastest point" changes as described on figure 3.

To check the standard separation at time t, we compute the distance between the two line segments modeling the aircraft positions and compare it to the standard separation at each time step of the simulation.

In the vertical plane, we use a cylindrical modeling (figure 3). Each aircraft has a mean altitude, a maximal altitude and a minimal altitude. To check if two aircraft are in conflict, the minimal altitude of the higher aircraft is compared to the maximal altitude of the lower aircraft.

[1]Ground track uncertainties will not be considered, as they do not increase with time and will be included in the standard separation

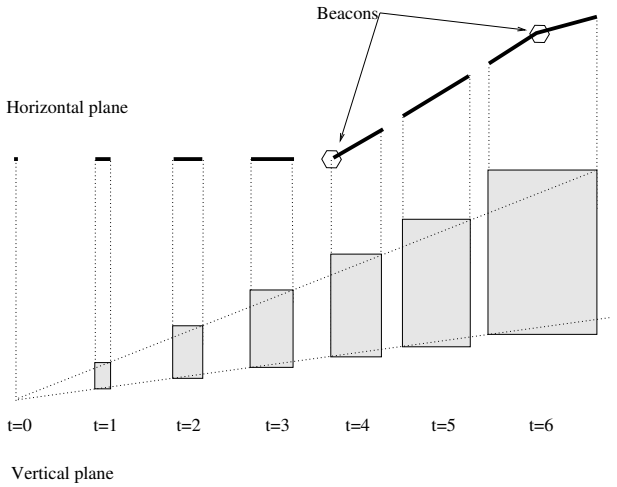


Figure 3. Modeling of speed uncertainties.

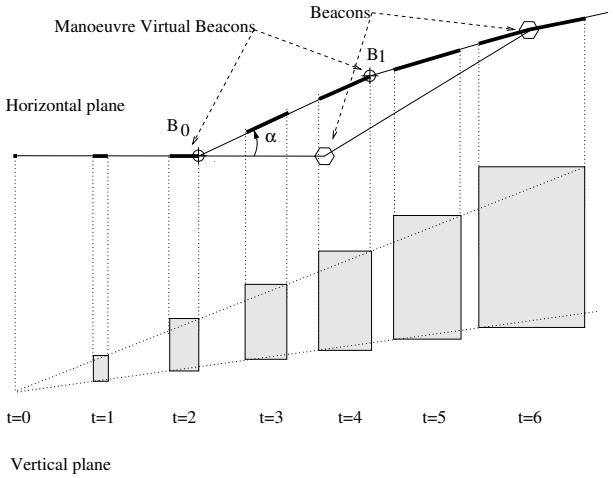


Figure 4. Horizontal maneuver modeling.

2) Clustering: After pair detection, P2 clusters conflicting aircraft. Each equivalence class for the relation "is in conflict with" becomes a cluster.

For example, if aircraft A and B are in conflict in the T_w window, and if B is also in conflict with C in the same time window, then A, B, C is the same cluster and will be solved globally by the conflict solver.

The conflict solver sends back to P2 maneuver orders for solving conflicts. Then P2 computes new trajectories for all aircraft and checks if new interferences appear. For example, if the new trajectory given to aircraft B to solve the conflict with A and C interferes with cluster D, E and with aircraft F, then A, B, C, D, E, F will be sent back to the problem solver as one conflict to solve.

The process will always converge. In the worst case, P3 will have to solve a very large cluster including all aircraft present in the next T_w minutes. However, this technique is usually efficient because a very large number of clusters can be solved very quickly in parallel.

Human controller's do not use any timestep, time window T_w or update period δ to deal with the traffic. We will see in the experimental results that a lot of calculation is done by the simulator but that only a few maneuvers calculated become effective. This is due to uncertainties that the simulator has to take into account. In real life, controllers also have to deal with uncertainties. Counting the number of maneuvers that never become effective can give a good measure of the impact of uncertainties on controllers workload.

II. Solver Modeling

A. Maneuver modeling

In the horizontal plane, classical maneuvers given to aircraft are heading deviation. In the simulator, 10, 20 or 30 degrees deviations will be allowed. The deviation starts on a virtual beacon created on the route (see figure 4). This beacon is defined by the position of the head of the segment at some time t_0. It ends on a second virtual beacon, position of the head of the segment at time t_1. An angle criteria is defined to find on which beacon the modified and initial routes should connect.

A maneuver will be determined by:

- t_0 which defines the first virtual beacon B_0.
- the deviation angle α.
- t_1 which defines the second virtual beacon B_1.

In the vertical plane, the aircraft trajectory is divided in 4 periods (figure 5):

- Climbing period. In this period, aircraft can be leveled at a lower than requested flight level to solve a conflict. The aircraft climb is stopped at flight level FL_0 and starts again on a virtual beacon B_1 as stated on figure 6. FL_0 and B_1 are defined by the position of the head of the uncertainty segment at time t_0 and t_1.
- Cruising period. When aircraft have reached their desired flight level, they may be moved

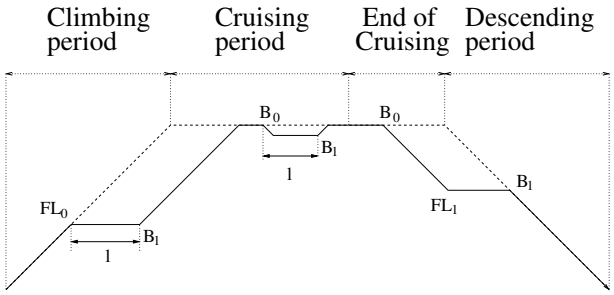


Figure 5. Vertical maneuver modeling.

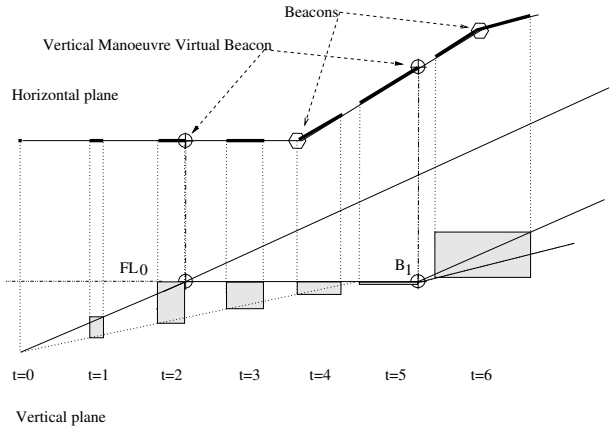


Figure 6. Vertical maneuver during the climbing period.

to the nearest lower level to resolve a conflict. Aircraft starts descending when reaching a virtual beacon B_0 and starts climbing at B_1 ($\alpha = 0$, B_0 and B_1 are defined by the position of the head of the uncertainty segment at time t_0 and t_1). An example of maneuver is represented on figure 7.

- End of Cruising period. When aircraft are about 50 nautical miles from beginning their descent to destination, they may be moved to a lower level to resolve a conflict. Aircraft start descending on B_0 and are leveled at FL_1 ($\alpha = 0$) (see figure 8). B_0 and FL_1 are defined by the position of the head of the uncertainty segment at time t_0 and t_1.
- Descending period. During this period no vertical maneuver is possible.

No maneuver will be simultaneously done in the horizontal and vertical plane. This model has the great advantage of reducing the size of the problem.

For a conflict involving n aircraft, the dimension of the search space is $3\,n$. This will allow us to solve

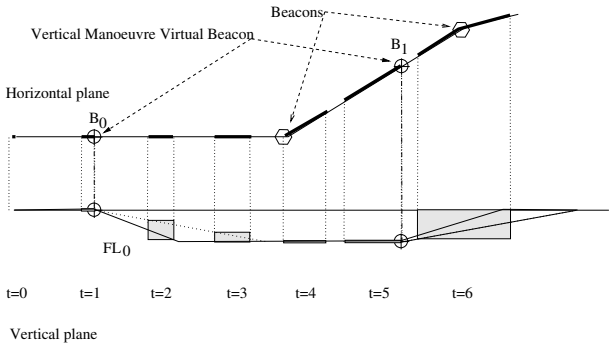


Figure 7. Vertical maneuver during the cruising period.

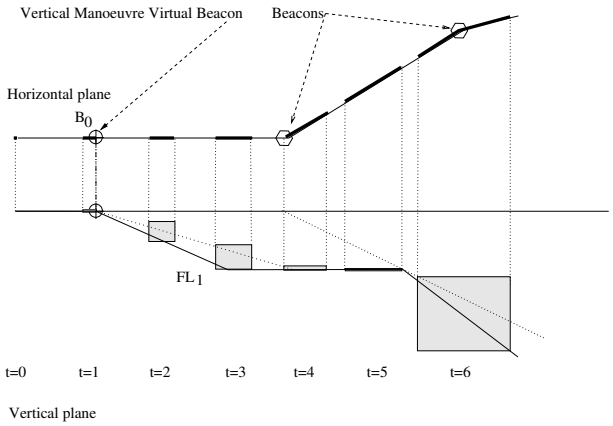


Figure 8. Vertical maneuver during the end of cruising period.

very difficult conflicts with many aircraft without investigating a large solution space.

B. Maneuver decision time

Because of uncertainties, some conflicts that are detected too early would not actually occur in the end. Consequently, deciding to move aircraft in such cases would be useless and could even generate other conflicts that would not occur if no maneuver had been decided. This explains why controllers do not solve conflicts too early. When there is no uncertainty, the earlier the maneuver is started, the lower the delay. However, if speed is not strictly maintained, the earlier the conflict is detected, the lower the probability it will actually happen. Thus, a compromise must be reached between the delay generated and the risk of conflict.

Because of uncertainties, maneuvers should be started as late as possible while still respecting aircraft constraints. First, it prevents the system from deciding unnecessary maneuvers. Second, an

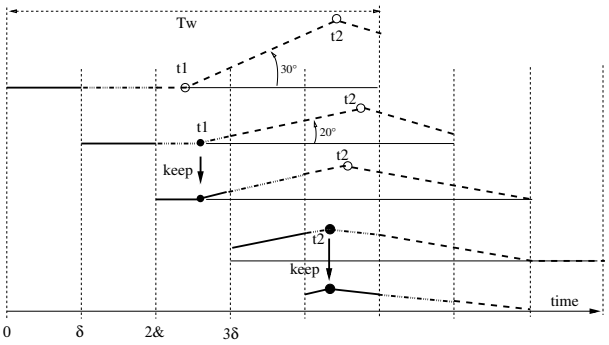


Figure 9. The model and real time optimization.

aircraft that is already maneuvering cannot make another maneuver before resuming its initial speed. The solver was modeled this way to keep the maneuvers simple to understand and execute. Starting maneuvers as late as possible increases the number of maneuverable aircraft.

Duration T_w can be changed, but must be at least equal to $2 \times \delta$. A good evaluation of T_w is difficult. With a perfect trajectory prediction, the largest T_w should be chosen. However, this is not true as soon as uncertainties are included in the model. A large value of T_w induces a large number of 1-to-1 conflicts, as the size of segments (modeling aircraft positions) grows quickly with time. Therefore, the conflict solver can become saturated.

When controllers are not too busy, they try to optimize the traffic and wait before giving a maneuver order that is not necessary. They sometimes prefer to solve a conflict ahead of time before it occurs in order to be able to take care of the rest of the traffic. It is very hard to reproduce this behaviour with an automatic solver, but we will show in this paper how the solver calculates many possible maneuvers that do not become effective but correspond to some control taskload.

C. A sliding forecast time window model

In order to limit the size of the problem and to be reactive to uncertainties, only the next T_w minutes of the flights are considered. T_w represents the lookahead time also called *forecast time window*. The situtation is revised every δ minutes with $\delta << T_w$. δ 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 δ minutes : current aircraft positions are updated which reduces uncertainties.

In figure 9, at $t = 0$, the aircraft trajectory cannot be modified before $t = \delta$ because any maneuver requires advance notice. Any maneuver that would occur between $t = \delta$ and $t = 2\delta$ would be kept as a constraint for the next optimization run (in the example, no maneuver is decided) because it will then be to close to the current time. In figure 9, the maneuver described on the first line resulting from an optimization at $t = 0$ (left turn of 30 degrees) is revised at time $t = \delta$ (left turn of 20 degrees) and then kept at time $t = 2\delta$. The end of the maneuver can be recalculated until the optimization starting at $t = 3\delta$.

Pilots should only be given maneuver orders that will not be modified; if no conflict occurs, no order will be given. In the example, the pilot will be notified of the beginning of the speed change at time δ and the end at time 3δ.

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. In order to solve conflicts with small speed adjustments, T_w needs to be large enough.

Figure 10 gives an example of two aircraft flying at the same level. In this example $T_w = 20$ minutes and $\delta = 4$ minutes. The speed uncertainty used is 5%. At time $t = 4$ minutes a potential conflict is detected and the solver calculates a maneuver (heading change) for one of the aircraft. Because the maneuver starts late enough, it is not sent to the aircraft. At time $t = 8$ minutes, the current positions of the aircraft are updated and the conflict is still detected. A new maneuver is calculated (heading change for the other aircraft), but still not sent to the aircraft because it starts late enough. At time $t = 12$ minutes, ther is a new update and the solver calculates a new maneuver (shorter than the previous one) but still late enough to remain an option. And at $t = 16$ minutes (see figure 10), after updating the current position of the aircraft the conflict has disappeared. This behaviour mimics the controllers' conflict detection task. When they are not too busy, controllers try to avoid unnecessary

manuevers but in dense traffic situations, they might however give a maneuver that solves immediately a potential conflict to concentrate on the rest of the traffic.

III. Solver Algorithm

Classical Evolutionary Computation (CEC) principles such as described in the literature [10], [11] is used in the solver.

A. 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 constraint in the fitness function.

The fitness function chosen is:

$$F = \frac{2\,n - nb_{man} - \sum_{i=1}^{n} \frac{\delta_i}{T_w}}{1 + n_{rc}}$$

where n is the number of aircraft, nb_{man} the number of maneuvers, δ_i the delay resulting from the maneuver of aircraft i and n_{rc} is the number of remaining conflicts.

The fitness function increases when the number of remaining conflicts and the delays decrease. It takes its values in $[0, 2\,n]$.

B. Crossover operator

The conflict resolution problem is partially separable as defined in [12], [13]. In order to increase the probability of producing children with a better fitness than their parents, principles applied in [12] were used. For each aircraft i of a population element, a local fitness F_i value is defined as follows:

$$F_i = \frac{2 - m_i - (\frac{\delta_i}{T_w})}{1 + nrc_i}$$

where nrc_i is the number of remaining conflicts involving aircraft i and $m_i = 0$ if aircraft i has no maneuver and 1 if it has a maneuver.

Figure 12 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.

C. 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 of 50%. We may 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 premature convergence, the sharing process introduced by Yin and Germay [14] 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:

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

D. 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.

IV. Experimental Results

CATS was used on a busy day of traffic (July 17^{th} 2010) in the French airspace. We were only interested in the upper airspace (above FL195). The number of flights in this airspace is 8870 for that specific day. Without any control maneuver, 2305

conflicts are detected (26% of the aircraft population). The mean time of flights is 57 minutes and the mean travelled distance in the french airspace is 394 nautical miles. Different experiments were done using CATS with different hypotheses.

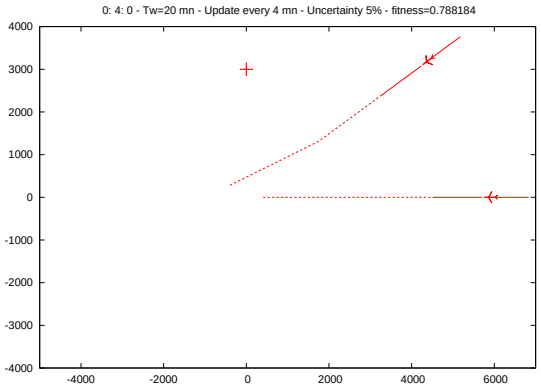
Tables I give the results of simulations for $T_w =$ 9 minutes and δ =2, 3 or 4 minutes for four sets of horizontal and vertical uncertainties (ranging from no uncertainty to 10% horizontally and 30% vertically). Column 5 gives the number of remaining conflicts. Column 6 gives the number of maneuvered aircraft and colum 7 the number of maneuvers sent to the pilots. Column 8 gives the number of maneuvers that were calculated by the solver but not sent to the pilots because they were supposed to start late enough to be revised at the next detection resolution process. Column 9 gives the number of potential conflicts detected in each scenario and column 10 gives a normalized number of potential conflicts by deviding the number of potential conflicts by $\frac{T_w}{\delta}$.

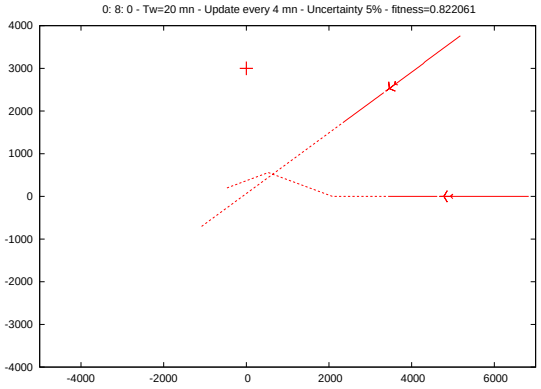
This table shows that the number of potential conflicts detected and the number of maneuvers not sent to pilots grow with uncertainty and also with the update frequency. For example, with 2% and 5% uncertainties, the normalized number of potential conflicts is multiplied by 1.6, with 5% and 10% uncertainties by 2.5 and with 10% and 30% uncertainties by 5.

When $T_w = 9$ minutes and $\delta = 4$ minutes the solver cannot solve every conflict which means that T_w needs to be big enough compared to δ to make the solver become efficient. The number of remaining conflicts increases with uncertainty. The fact that the number of maneuvers sent to pilots increases with δ reflects that updating the situation is very important. This explains why the detection resolution process is a permanent task for controllers. The high number of maneuvers that are never sent to pilots can also give an idea of the complexity of the detection task. Indeed, many potential conflicts disappear after updating aircraft positions. However, both an automatic solver or a human controller have to deal with these potential conflicts. The solver calculates maneuvers starting late enough to be updated in the future (unsent maneuvers). Human controllers have to keep the traffic safe and send as little unnecessary maneuvers as possible. These experimental results show that the number of maneuvers sent to pilots is a small portion of the total number of maneuvers calculated (for example with $T_w = 9$ minutes, $\delta = 2$ minutes and a $(5\%, 10\%)$ uncertainties, only 30% of the maneuvers are sent to pilots). This suggests that the human controllers taskload is probably more dedicated to conflict detection than to conflict resolution.

Table II gives the results of the simulation for $T_w = 12$ minutes. The same comments can be made on this table as on table I. The number of maneuvers sent to pilots decreases with the update frequency and increases with uncertainties. For example, with no uncertainty and $\delta = 2$ minutes, the number of potential conflicts is 10166, the number of maneuvers sent to pilots is 2095 and the number of maneuvers unsent is is 6325. When the uncertainty increases to 5% horizontally and 10% vertically the number of maneuvers sent to pilots doubles (4010), the number of unsent maneuvers is multiplied by 2.5 (16087) and the number of potential conflicts is multiplied by 2.6 (26860). With $T_w = 9$ min and $\delta = 2$ minutes the number of maneuvers necessary to solve every conflict is multiplied by 3 when dealing with 10% and 30% uncertainty and the number of maneuvers calculated but not sent is multiplied by almost 4 and the number of potential conflicts is multiplied by almost 5. This is not surprising because unsent maneuvers deal with conflicts that might occur later and are more influenced by uncertainties. This suggests that uncertainty has a great influence on the controller's taskload and that a good Trajectory Prediction tool could probably help controllers to deal with uncertain conflicts and reduce their detection monitoring task.

Results also show that uncertainties have a big influence on the number of maneuvers needed to solve the conflicts. Controllers are used to saying that they often have to solve conflicts that might never happen because of uncertainties. It appears that an automatic tool would encounter the same issue. These experiments show how an automatic solver can model the taskload of human controller. It first needs to calculate many maneuvers to solve potential conflicts but tries to only make decisions that are absolutely necessary.





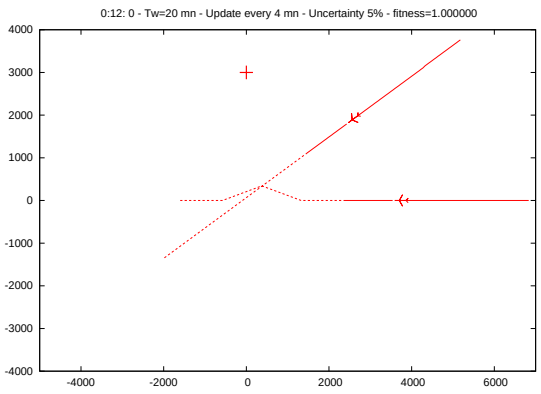
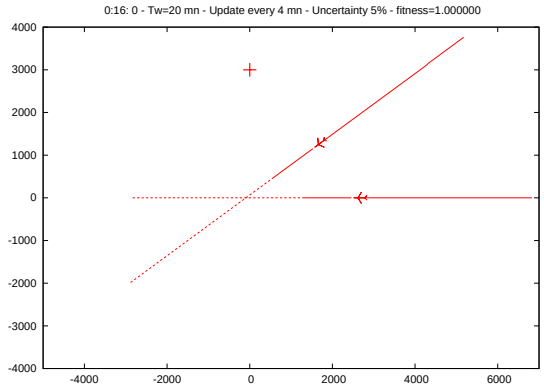


Figure 10. Two aircraft potential conflict at time t=4,8,12 minutes.



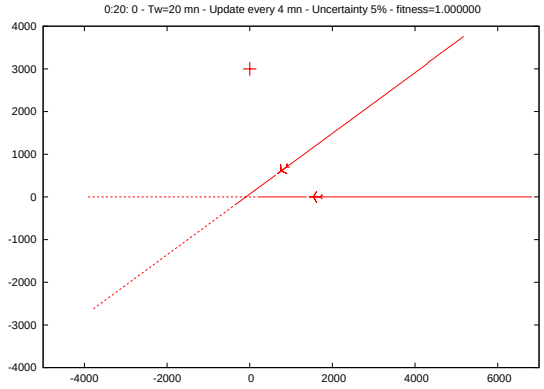


Figure 11. No more potential conflict at time t=16 and 20 minutes.

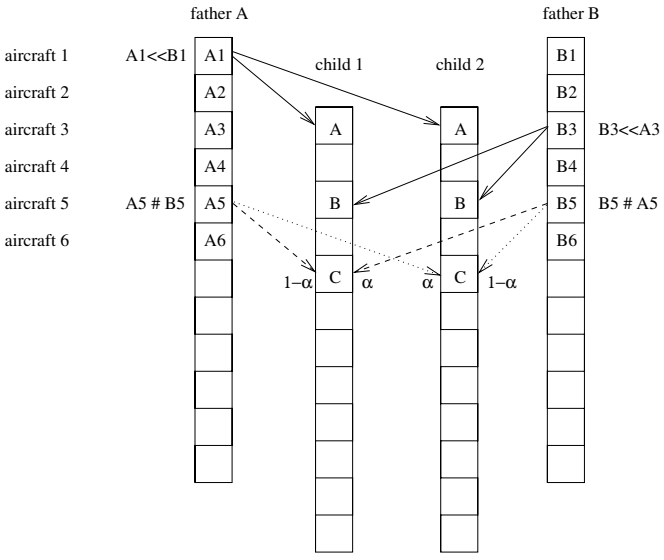


Figure 12. Crossover operator

T_w	δ	horiz uncert	vert uncert	remain conflict	maneuvered acft	maneuvers sent	maneuvers not sent	pot confs	norm pot confs
9	2	0	0	0	1925	2395	4208	7620	1693
9	3	0	0	0	2095	2667	1201	4627	1542
9	4	0	0	140	2242	2997	120	3729	1657
9	2	2	5	0	2540	3352	7117	12421	2760
9	3	2	5	0	2916	4056	2079	7730	2577
9	4	2	5	169	3296	5037	112	6284	2793
9	2	5	10	0	3297	4551	10281	18742	4165
9	3	5	10	0	3934	5929	2771	12037	4012
9	4	5	10	273	4448	7510	171	9939	4417
9	2	10	30	0	4657	7181	16064	35484	7885
9	3	10	30	52	5675	10278	3806	24198	8066
9	4	10	30	387	6225	12521	309	19186	8527

TABLE I

$T_w = 9$ MINUTES:NUMBER OF REMAINING CONFLICTS,NUMBER OF MANEUVERED AIRCRAFT AND GIVEN AND CALCULATED BUT NOT GIVEN MANEUVERS FOR DIFFERENT δ VALUES AND DIFFERENT UNCERTAINTIES.

T_w	δ	horiz uncert	vert uncert	remain conflict	maneuvered acft	maneuvers sent	maneuvers not sent	pot confs	norm pot confs
12	2	0	0	0	1756	2095	6325	10166	1694
12	3	0	0	0	1914	2338	2942	6304	1576
12	4	0	0	0	2058	2496	1158	4355	1452
12	2	2	5	0	2384	2924	10887	17200	2867
12	3	2	5	0	2721	3457	5153	10699	2675
12	4	2	5	0	3040	4125	2087	7771	2590
12	2	5	10	0	3127	4010	16087	26860	4477
12	3	5	10	0	3600	4840	7902	17444	4361
12	4	5	10	0	4188	6160	2846	12637	4212
12	2	10	30	0	5047	7809	24634	51917	8653
12	3	10	30	0	5559	9164	11349	37738	9434
12	4	10	30	65	6308	11680	3771	29729	9910

TABLE II

$T_w = 12$ MINUTES:NUMBER OF REMAINING CONFLICTS,NUMBER OF MANEUVERED AIRCRAFT AND GIVEN AND CALCULATED BUT NOT GIVEN MANEUVERS FOR DIFFERENT δ VALUES AND DIFFERENT UNCERTAINTIES.

The number of potential conflicts detected and the number of maneuvers needed to solve them give a better idea of the conflict detection complexity than the final number of actions that the controller really had to take.

V. Conclusion

The CAT Simulator cannot reproduce human controllers' behaviour because it uses the computer's power to detect and build solutions in advance and validate them when necessary. Human controllers rely on their expertise and experience on specific sectors to take the right action at the right moment. It is very difficult to reproduce the human behaviour. Researchers studying chess have managed to build softwares able to beat the best players by using the computer's ability to calculate very fast but not by imitating the players' way of thinking. However the amount of calculation necessary to replace the controller's decision on fast time simulation probably gives a better idea of the controller's taskload than the result of the calculation itself (i.e. the number of conflicts detected or maneuvers sent to pilots).

In this article we tried to show how the CATS solver can reproduce some of the controller's task that does not appear by simply counting the number of actions that were taken.

The controller's permanent detection process is modeled by a sliding forecast time window that is updated every δ minutes. The smaller δ is, the more efficient the solver is.

Uncertainties are costly regarding the number of potential conflicts and unsent maneuvers that are detected. The controllers detection taskload is hard to measure because only a small proportion of conflicts end up by a maneuver order. The number of potential conflicts detected and unsent maneuvers can give a good idea of the extra amount of work controllers have to deal with because of uncertainties.

In order to complete the work presented in this paper it would be interesting to correlate the number of potential conflicts or unsent maneuvers with the human taskload. This is a difficult task because it would require defining scenarios on which we could both have measures of the human taskload and CATS simulation results. Exising data in France involve TMA (Terminal Control Area) traffic for wich CATS is not yet adapted. We look forward to work on this in the future.

References

[1] P. Averty, S. Athenes, C. Collet, and A. Dittmar, "Evaluating a new index of mental workload in real control situation using psychophysiological measures," *21th DASC*, 2002.

[2] C. Collet, P. Averty, A. Dittmar, and E. Vernet-Maury, "Workload in air traffic controllers estimated by autonomic nervous system activation and subjects self estimation," *Psychophysiology*, p. 37, 2000.

[3] D. Gianazza and K. Guittet, "Selection and evaluation of air traffic complexity metrics," in *25th DASC*, 2006.

[4] N. Durand, J.-M. Alliot, and G. Granger, "A statistical analysis of the influence of vertical and ground speed errors on conflict probe," in *Proceedings of the 4th USA/Europe R and D Seminar*, 2001.

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

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

[7] ——, "FACES: a Free flight Autonomous and Coordinated Embarked Solver," in *2 ST U.S.A/EUROPE ATM R & D Seminar*, December 1998.

[8] ——, "Token allocation strategy for free-flight conflict solving," in *IJCAI'01*, 2001.

[9] N. Durand, J. Alliot, and G. Granger, "Faces: a free flight autonomous and coordinated embarked solver," *ATC Quarterly*, 1999.

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

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

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

[13] 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.

[14] 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.