An optimizing conflict solver for ATC

Nicolas Durand, Jean-Marc Alliot, Olivier Chansou

An optimizing conflict solver for ATC

Nicolas Durand Jean-Marc Alliot Olivier Chansou

CENA* ENAC† CRNA‡

October 13, 1995

Introduction

As traffic keeps increasing, En Route capacity, especially in Europe, becomes a serious problem. Aircraft conflict resolution, and resolution monitoring, are still done manually by controllers. Solutions to conflicts are empirical and, whereas aircraft are highly automated and optimized systems, tools provided for ATC control are very basic, even out of date. If we compare the current capacity and the standard separation to the size of controlled space, the conclusion is easy to draw: while ATC is overloaded, the sky is empty. It must be noticed that enhancing En Route capacity does not require optimal resolution of aircraft conflicts.

The need for an automatic problem solver is also a serious concern when addressing the issues of free flight. It is still very much unclear how conflicts will be solved in free flight airspace. Human controllers highly rely on standard routes and traffic organization for solving conflicts; they are much quickly overloaded when controlling aircraft flying on direct routes. Free flight traffic, with a completely unorganized structure, might require automated, computer based, solvers. Moreover, one of the aim of free flight is to give aircraft optimal trajectories, whereas using ACAS techniques is certainly not optimal, and should only looked upon as a last issue system.

In this paper, we present an optimal problem solver, based on a stochastic optimization technique (genetic algorithms). It builds optimal resolution for complex conflicts and also computes a large number of nearly optimal resolutions that do not violate separation constraints.

Part 1 of the paper introduces the problem solver, its constraints and goals. Modelling is discussed in part 2. Part 3 introduces genetic algorithms techniques and the coding of the problem. Part 4 presents different examples of resolution of very complex test problems.

*Centre d'Etudes de la Navigation Aérienne

†Ecole Nationale de L'Aviation Civile

‡Centre Régional de la Navigation Aérienne

In part 5 we present the complete ATC simulator, conflict detector and cluster builder used to benchmark the problem solver on real traffic; we also discuss weaknesses of the system and possible improvements.

1 Conflict resolution

1.1 Problem solvers

Conflict resolution is a very complex mathematical problem involving trajectory optimization and constraints handling. This problem has two faces: find automatically solutions to the conflicts, and find the *optimal* solution regarding delays. There have been many attempts to reach these objectives, automation and optimization. However, most of the time, these two objectives are confused:

- AERA 3 [NFC+83, Nie89b, Nie89a] considers optimum results in the "Gentle-Strict" function for a two aircraft conflict, but the "Maneuver Option Manager" only seeks after acceptable solutions and does not focus on the optimum.
- Karim Zeghal [Zeg94], with reactive techniques for avoidance, gives a solution to the problem of automation which is robust to disturbance, but completely disregards optimization. Furthermore, the modelling adopted implies a complete automation of both on board and ground systems and requires speed regulations which can not be handled by human pilots and would probably be very difficult to apply to current aircraft engines without damaging them.
- ARC-2000 [K+89, FMT93] optimizes aircraft trajectories using 4 dimensional cones and priority rules between aircraft. Optimum is not reached. Moreover the system relies on the availability of FMS-4D for all aircraft[1].
- A first approach to conflict resolution by genetic algorithms[2] was done by Alliot and Gruber [AGS93]; more advanced results were presented in [DASF94b, DAN96]. An other approach using also genetic algorithm has been tried by Kemenade, Hendriks, Hesseling and Kok [vKHHK95].

1.2 Specifications of the system

The main idea behind the design of the solver was to be as close as possible to the current ATC system.

Constraints: the solver has to handle the following constraints:

[1] It must be noted that only the ARC-2000 system has been tested on real traffic.

[2] It must be noted that genetic algorithms were also applied to airspace sectoring with promising results [DASF94a].

- Conflict free trajectories must respect both aircraft and pilot performances. Considering the evolution of ATC toward automation [DAM93], trajectories must remain simple for controllers to describe as well as for pilots to understand and follow.
- Trajectories must take into account uncertainties in aircraft speed[3] due to winds or turbulence.
- Maneuvers orders must be given with an advance notice to the pilot. When a maneuver has begun, it must not be called into question. A maximum of one maneuver per aircraft should be forecasted for the next twenty minutes.
- If possible, conflicts must be solved horizontally for comfort and economical reasons, especially when aircraft are leveled.

Goals: We want to achieve the following goals:

- find conflict free and optimal trajectories
- compute these trajectories in real time
- find many different optimal or nearly-optimal solutions, so that in a transition phase to automation, this tool could be used to assist a human operator

2 Modelling

In this paper we will only give resolution orders in the horizontal planes. It could and will be extended to $3D$ conflict resolution. However, detection will be done in $3D$.

2.1 Theoretical results

Optimal Command Theory with State Constraints developed by Bryson and Ho [BH75], supplementary conditions exposed by Kreindler [E.K82], Bryson, Denham and Dreyfus [JWS63] lead to the following conclusions exposed by Durand, Alech, Alliot and Schœnauer in [DAAS94]. For a conflict resolution involving two aircraft: at the optimum, as long as the standard separation constraint is not saturated, aircraft fly in straight line. When saturating, aircraft start turning, and as soon as the separation constraint is freed aircraft fly straight again. This result can easily be extended to the case of n aircraft, with $n \geq 2$. When moving only one aircraft, it can be proved (see [Dur94]) that trajectories are regular, they do not include discontinuous points and the minimum increase x (in percentage) of the trajectory length is described by the following implicit equation:

$$2 + x = \frac{1}{1 + \frac{d}{\sqrt{2-d^2}}} + \frac{2 + 2\,x + x^2}{2\,\left(1 + x - \frac{d}{\sqrt{2+2\,x+x^2-d^2}}\right)} + \frac{d}{2}\log\left(\frac{(1+d)\,\left(\sqrt{2+2\,x+x^2-d^2}+1\right)}{(1+x-d)\,\left(\sqrt{2-d^2}+1\right)}\right)$$

[3] Aircraft ground speeds can not be forecasted precisely because of winds, and radar precision. Moreover, models to forecast aircraft ground speeds are not reliable enough. Consequently, uncertainty on speeds have to be introduced in the model. We will discuss these issues in part 5.

In this equation d is the standard separation divided by the initial distance to the trajectory cross, speeds are equal and normalized, the trajectory angle is 90 degrees.

This equation can be generalized to different speeds and angles. Numerical resolutions shows that the length of the conflict free trajectory increases when:

- the angle of incidence between the two aircraft decreases.
- the speed ratio gets close to 1.
- aircraft are getting closer to the conflict point.

It can also be mathematically proved (see [Dur94]), that if aircraft parameters (speed and heading) are constant at intervals, and if aircraft trajectories don't loop, the set of conflict free trajectories has two connected components. In one of the two sets, one of the aircraft always lets the other one on its right side, whereas in the other set, it lets it on its left side.

The previous mathematical study leads naturally to approximate the conflict free trajectories by a turning point trajectory (figure 1).

When only one of the two aircraft turns, it has been shown that the turning point approximation lengthens the optimal trajectory by less than 1% if distance between the aircraft and the conflict point is greater than two standard separations and the angle of incidence between trajectories is greater than 30 degrees. It can also be proved that the offset modelling (figure 1), which moves an aircraft to put it on a parallel route, is worse. However, it has the advantage to linearize the separation constraints. The offset is thus very easy to compute, but separation constraints must be checked during maneuvers and the complexity of the problem remains. For n aircraft, $2^{\frac{n\,(n+1)}{2}}$ linear programs must be solved: if $n = 6$, 2097152 linear programs must be solved, if $n = 7$, the latter number grows to more than 268 million (see [MDA94]). Moreover, the offset modelling requires one more maneuver for the pilot. Anyway this type of maneuver is very useful to solve conflicts between overtaking aircraft and we will still consider it along with the turning point modelling.

2.1.1 Separating aircraft

Nowadays, aircraft are able to follow a given route at a given altitude, but their speed can not be precisely predicted. So, controllers often adopt a wait-and-see attitude, in order to be sure that the conflict will really occur, before solving it. In this paper, we will assume that there is an error about the aircraft future location because of ground speed uncertainties. The uncertainties on climbing and descending rates are even more important. The conflict free trajectory must be robust regarding these uncertainties. So, the aircraft will be represented by a point at the initial time. The point will become a line segment in the uncertainty direction, the speed direction in our case. To check the standard separation at time t, we compute the distance between the two segments modelling the aircraft positions and compare it to the standard separation (see figure 2).

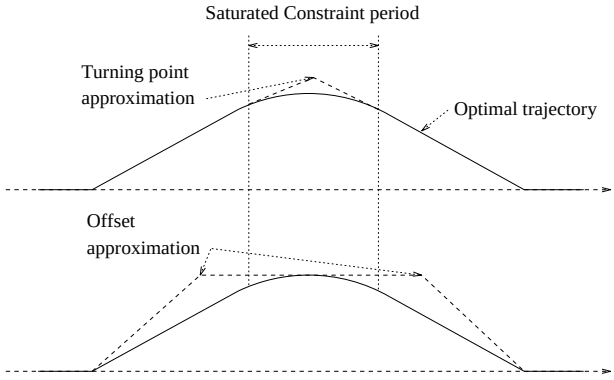

Figure 1: Turning point and offset approximation.

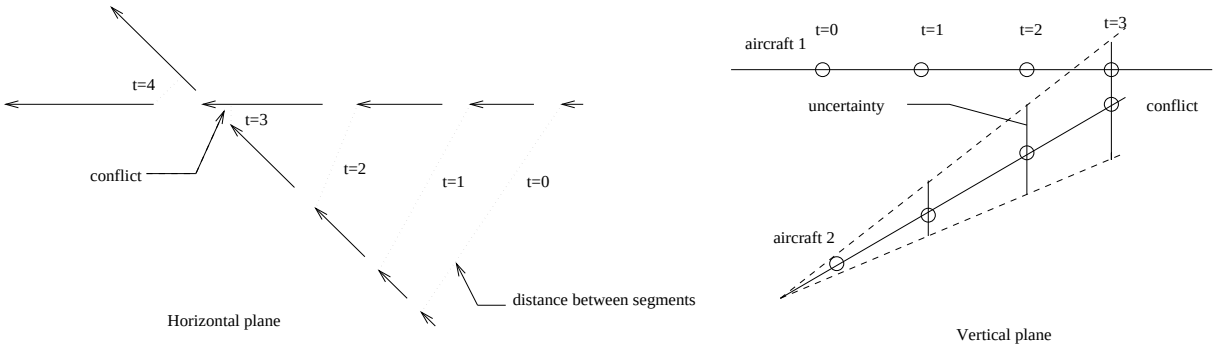

Figure 2: Modelling of speed uncertainties.

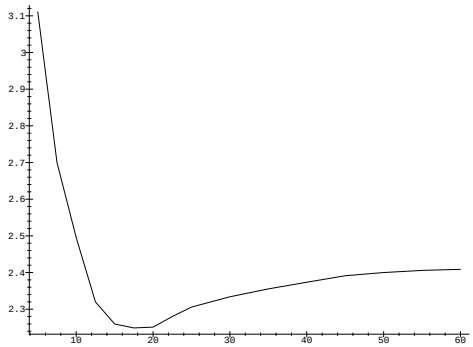

Figure 3: Resolution cost expectation for different anticipation times.

In the vertical plane, we use a conical modelling (see left part of figure 2). Each plane has a mean altitude, a maximal altitude and a minimal altitude. To check if two planes are in conflict, the minimal altitude of aircraft 1 (resp 2) is compared to the maximal altitude of aircraft 2 (resp 1).

2.1.2 Maneuver decision time

Because of uncertainties, a conflict may be detected twenty minutes before it should occur and finally may not happen. Consequently, deciding to move an aircraft in that case 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. With the turning point modelling, when there is no uncertainty, the earlier the maneuver is started, the lower the delay. However, if speed is not strictly respected 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. Let's imagine a conflict between 2 aircraft flying at the same speed (400 knots) and converging at a right angle (uncertainty on speed is 5 per cent and the standard separation is 10 nautical miles). We can define the resolution cost expectation of the conflict as the product of the resolution cost and the probability of conflict for a given anticipation time. Figure 3 shows that for this example, the best maneuver decision time is about 20 minutes before the conflict. This results confirms that "the earlier the better" is not the best tactic when uncertainties are introduced in the model.

For the offset modelling, the resolution cost is not depending on the anticipation time so that "the later, the better" is a good strategy.

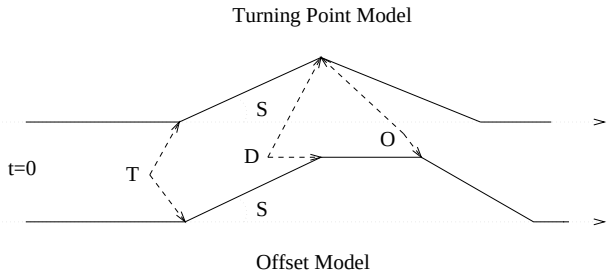

Figure 4: The Model.

2.2 Choosing the model

Both offset modelling and turning point modelling must be kept: for overtaking aircraft, offsets are more efficient whereas for other conflicts, turning points are more efficient. However, if we do not want to call into question previous maneuvers and be able to solve very large conflicts, we must try to start maneuvers as late as possible with respect to the aircraft constraints. This argument is enforced by the fact that we allow aircraft to have large uncertainties on their speeds. The turning point angle will be 10, 20 or 30 degrees. The previous elements lead us to choose the following model (figure 4). A maneuver will be determined by:

- the maneuver starting time T.
- the turning point time D.
- the offset ending time O (if no offset, this value will be equal to D).
- the deviation angle S.

This model has the great advantage to reduce the size of the problem. For a conflict involving n aircraft, the dimension of the research space is $4n$. This will allow us to solve very difficult conflicts with many aircraft without investigating a too large space.

2.3 Real time optimization

The model introduced above is simple enough to be used in a real time optimization program[4]. Let's consider a conflict involving n aircraft and let's choose a time step (δ minutes for example). Let's imagine we want to recompute all trajectories every δ minutes. During the optimization time, aircraft are flying and must know if they must change their route or not. Consequently, for each aircraft, at the beginning of the current optimization, trajectories are determined by the previous run of the problem solver and can not be changed for the next δ minutes; afterwards, an offset can be shortened or turned into a turning point, a turning point can become an offset, etc.

[4]Simulating 20 minutes of aircraft flight and checking standard separations is very long. Consequently, a simple modelling is necessary to achieve a real time optimization.

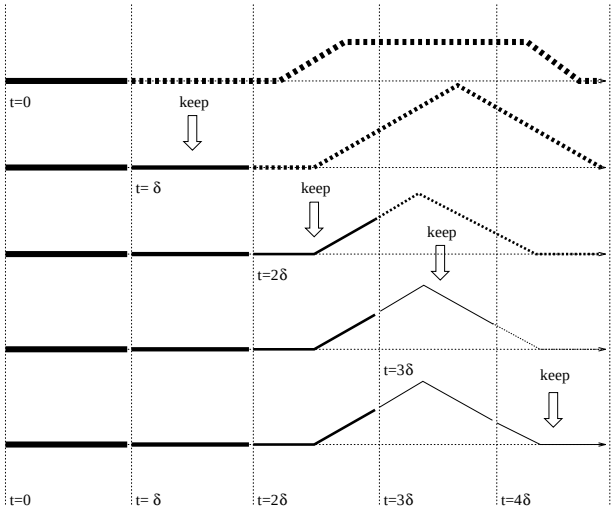

Figure 5: The model and real time optimization.

For example the first trajectory of figure 5, at $t = 0$, can not be modified before $t = \delta$. At the end of the first optimization run, at $t = \delta$, the current position of the aircraft is updated. The maneuver that occured between $t = \delta$ and $t = 2\delta$ is kept as a constraint for the second optimization run (on the example, no maneuver is decided).

On the above example, we can see that the maneuver described on line 2 (resulting of an optimization at $t = \delta$) is more penalizing than the maneuver described on line 3 (resulting of an optimization at $t = 2\delta$). This phenomenon occurs because of uncertainties. If uncertainties on speed are important, having a small δ will be very helpful to minimize the resolution costs in real time situation.

2.4 Complexity of the problem

The complexity of the problem is exposed by Medioni, Durand and Alliot in [M94]. Let's consider a conflict between two aircraft. We can easily prove that the minimized function is convex, but the set of conflict free trajectories is not. It is not even connected. If trajectories don't loop, the set of conflict free trajectories has two connected components. For a conflict involving n aircraft there may be 2^n connected components in the free trajectory space which strongly suggests that any method which requires exploring every connected component is NP. It is important to note that this complexity is independent of the modelling chosen. The offset modelling seems to be very attractive, because it linearizes constraints. Nevertheless, each constraint multiplies by two the number of linear programs to solve. Our problem involves $\frac{n\,(n-1)}{2}$ constraints. Moreover, linearizing the minimized function, multiplies by 2^n the number of linear programs to solve (we minimize the sum of each aircraft offset which may be positive or negative). Finally, we will have to solve $2^{\frac{n\,(n+1)}{2}}$ linear programs, each one involving $\frac{n\,(n+1)}{2}$ linear constraints. For $n = 5$, we have to solve 32768 linear programs with 15 constraints in each program.

2.5 A global optimization problem

Using classical methods, such as gradient methods for example, becomes useless for our problem, because of the arbitrary choice of the starting point required by these methods. Each connected component may contain one or several local optima, and we can easily understand that the choice of the starting point in one of these components can not lead by a classical method to an optimum in another component. We can thus expect only a local optimum. Practical attempts done on LANCELOT *Large And Nonlinear Constrained Extended Lagrangian Optimization Techniques* [CGT92] have confirmed this problem and high-lighted others. Convergence is very slow, particularly when introducing a speed constraint. Solving a five aircraft conflict without vertical offset takes a week on a Sun work-station. This approach is not efficient for a real time trajectory planning. The offset modelling linearizes the constraints but the problem remains so combinatorial that we can not expect to find the global optimum efficiently. Moreover, the separation during maneuvers must be checked afterwards. This experiment strongly suggests that classical methods are not well adapted to our problem.

3 Coding

3.1 Genetic Algorithms

Genetic algorithms were initially developed mainly by John Holland [Hol75] in the sixties. Genetic algorithms are stochastic optimization techniques that mimic natural evolution. Figure 6 describes the main steps[5] of GAs: first a population of points in the state space is randomly generated. Then, we compute for each population element the value of the function to optimize, which is called *fitness*. Then the *selection process* reproduces elements according to their fitness. Afterwards, some elements of the population are picked at random by pairs. A *crossover operator* is applied to each pair and the two parents are replaced by the two children generated by the crossover. In the last step, some of the remaining elements are picked at random again, and a *mutation operator* is applied, to slightly modify their structure. At this step a new population has been created and we apply the process again in an iterative way. The different steps are detailed in the following.

3.1.1 Selection

A method called "Stochastic Remainder Without Replacement Selection" [Gol89] is classically used. First, the fitness f_i of the n elements of the population is computed, and the average $a = \sum f_i/n$ of all the fitnesses is computed. Then each element is reproduced p times in the new population, with $p = truncate(n \times f_i/a)$. The population is then completed using probabilities proportional to $f_i - p\,a/n$ for each element.

[5] We are using classical Genetic Algorithms and Evolutionary Computation principles such as described in the literature [Gol89, Mic92].

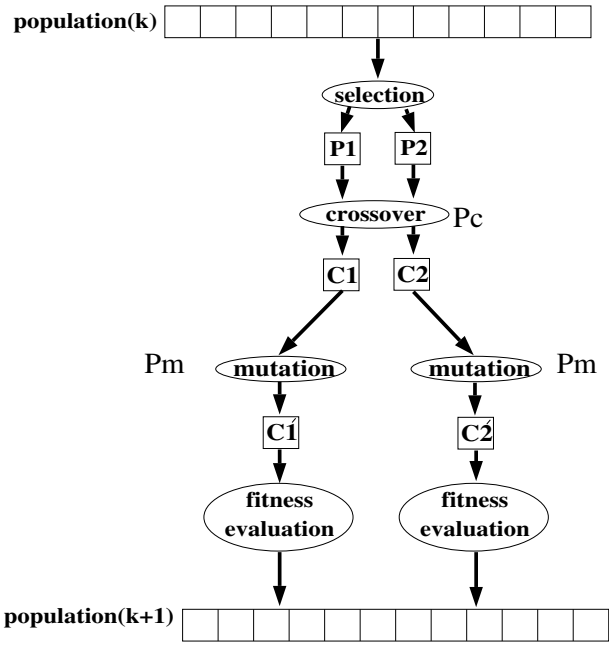

Figure 6: GA principle

3.1.2 Crossover

The crossover operator randomly chooses two elements of the population (called parents), and combines them to create two children that will replace them. Usually, between 25% to 75% of the population is crossed. Several crossover operators can be used:

- the slicing crossover takes one part of the variables of parent 1 (resp 2) and completes it with the other part of the variables of parent 2 (resp 1) to create child 1 (resp 2).
- the barycentric crossover is used with real variables and recombines two parents by choosing randomly $\alpha \in [-0.5, 1.5]$ and creating child 1 (resp child 2) as the barycenter of $(parent_1, \alpha)$ (resp $(parent_1, 1-\alpha)$) and $(parent_2, 1-\alpha)$ (resp $(parent_2, \alpha)$).

Most of the time, crossover operators are adapted to the problem so that they generate children that respect constraints. Heuristics are sometimes used to improve their efficiency. However, the crossover operator must not be too deterministic, otherwise the algorithm may converge quickly to a local optimum and get stuck.

3.1.3 Mutation

The mutation operator randomly chooses an element of the population and slightly modifies it.Between 0.1% to 15% of the population is usually mutated. Depending on the types of variables used, the mutation operator can change a bit of a string or add a noise to a real variable. The goal of this operator is to generate new elements that explore the search space.

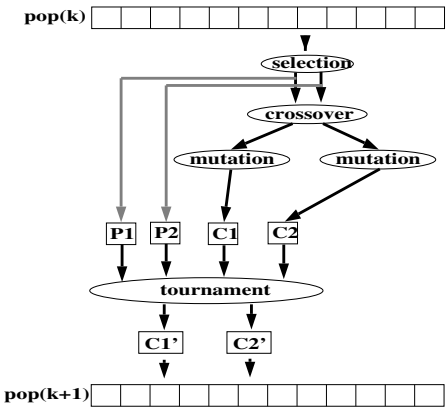

Figure 7: GA and SA mixed up

3.1.4 Simulated Annealing Tournament

GA are not a magical tool, coming out of nowhere. They are closely related to random search techniques. For example, random walk in a search space is a GA with only one population element and no crossover. As for random walk, GA can be improved by including a Simulated Annealing process after applying the operators [MG92]. For example, after applying the crossover operator, we have four individuals (two parents $P1$,$P2$ and two children $C1$,$C2$) with their respective fitness. Afterward, those four individuals compete in a tournament. The two winners are then inserted in the next generation. The selection process of the winners is the following: if $C1$ is better than $P1$ then $C1$ is selected. Else $C1$ will be selected according to a probability which decreases with the generation number (any cooling scheme used in simulated annealing can be used). At the beginning of the simulation, $C1$ has a probability of 0.5 to be selected even if its fitness is worse than the fitness of $P1$ and this probability decreases to 0.01 at the end of the process. A description of this algorithm is given on figure 7. Tournament selection brings some convergence theorems from the Simulated Annealing theory. On the other hand, as for Simulated Annealing, the (stochastic) convergence is ensured only when the fitness probability distribution law is stationary in each state point [AK89].

3.2 Coding our problem

An example of chromosome is given in figure 8. Each time value represents is coded by a positive integer ($T_i \leq D_i \leq Oi$). The deviation angle can be -30, -20, -10, 0, 10, 20 or 30.

At each new step, some data must be given:

1. The duration of the optimization.

2. The anticipation time.

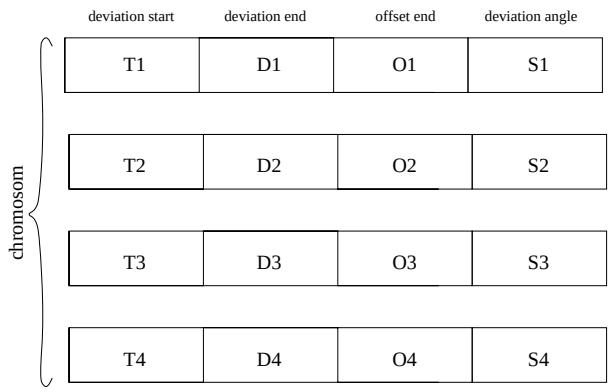

Figure 8: Structure of the chromosome

3. The time step used during the simulation.
4. The horizontal separation, in Nm.
5. The position of each aircraft.
6. The heading of each aircraft.
7. The speed of each aircraft.
8. The uncertainty on each aircraft speed (it can depend on its characteristics, its equipment and on the weather conditions).

Other global data are required by the Genetic Algorithm such as the number of generations, the number of elements, the percentage of elements to cross and the percentage of elements to mutate.

3.3 Computing the fitness

One of the main issues is to know how to compute the *fitness* of a chromosome. We have a poly criteria problem to solve, in fact the following criteria have to be matched together to give us a single fitness function:

- The delay due to a deviation for each aircraft must be as small as possible.
- However, the number of aircraft deviated and the total number of maneuvers must be as low as possible. Thus, instead of sharing the global delay on all the aircraft, some aircraft will support a part of the delay and others not. The sharing process described above is very important to give different equivalent solutions.
- The maneuver duration for an aircraft must be as short as possible so that the aircraft is freed as soon as possible for another maneuver.

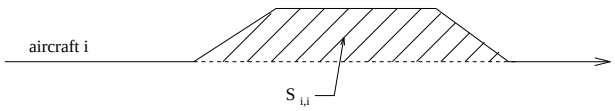

Figure 9: Maneuver Occupancy Surface

- The trajectories must handle the separation constraints.

Instead of considering a global fitness value that takes into account the different lengthenings of the trajectories and the conflicts between the aircraft, we keep in a n^2 sized matrix F (where n is the number of aircraft) these values. If $i \neq j$, $F_{i,j}$ measures the conflict between aircraft i and j. It is set to 0 if no conflict occurs and increases with the seriousness of the conflict. $F_{i,i}$ measures the lengthening of aircraft i trajectory. This fitness matrix contains much more information than the previous scalar global fitness and this will allow us to define more deterministic crossover and mutation operators.

To take into account both the delay and the maneuver duration, we decided to measure the maneuver occupancy surface S (figure 9). This value increases with both the delay and the maneuver duration. It is interesting to note that controllers currently use this notion of maneuver occupancy surface. To minimize the number of maneuvers, add to S the number of maneuvers N multiplied by a coefficient k ($N = 0$ if no maneuver, 2 if it is an turning point and 3 if it is an offset).

At each time step t, we compute $C_{t,i,j}$ as the difference of the standard separation and the distance between the segments i and j describing aircraft i and j position at time t. These values are added and give a measure of the conflict between i and j. So, the fitness matrix is computed as follows:

$$\begin{aligned} F_{i,i} &= S_i + k\, N_i \\ F_{i,j} &= \sum_{t=0}^{total\,time} (C_{t,i,j}) \end{aligned}$$

It is obvious that the fitness matrix is symmetrical. A triangular matrix could as well be used. We can now define a global scalar fitness as follow:

$$\begin{aligned} \exists (i,j),\, i \neq j,\, F_{i,j} \neq 0 \;\Rightarrow\; F &= \frac{1}{2 + \sum_{i \neq j} F_{i,j}} \\ \forall (i,j),\, i \neq j,\, F_{i,j} = 0 \;\Rightarrow\; F &= \frac{1}{2} + \frac{1}{1 + \sum_i F_{i,i}} \end{aligned}$$

This fitness function guarantees that if a chromosome value is larger than $\frac{1}{2}$, no conflict occurs. If a conflict remains the fitness does not take into account the delays induced by maneuvers.

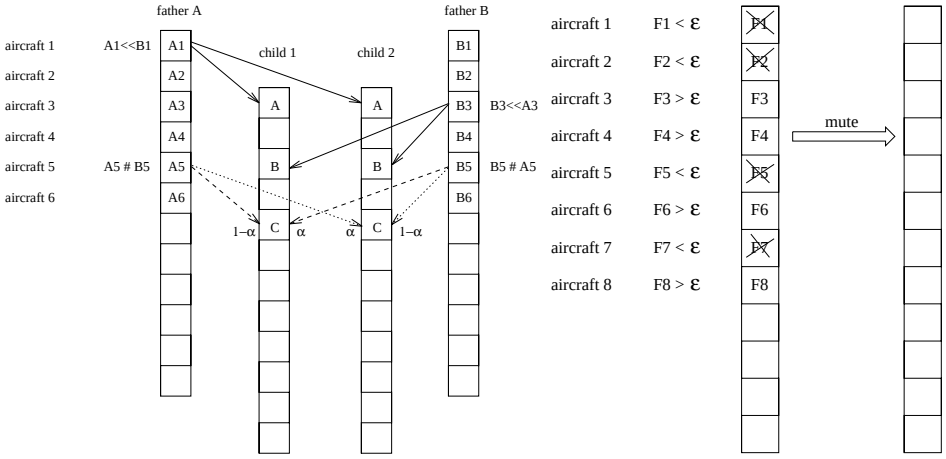

Figure 10: Adapted crossover and mutation operators

3.4 Improvements

3.4.1 Adapted Crossover and Mutation

GAs seem very powerful because they do not require much information on the fitness function. However, classical operators used by Gruber, Alliot and Schoenauer in [AGS93] did not give good results. Moreover, we know much about the fitness function and it can be very useful to use this information to create adapted crossover and mutation operators. Durand, Alliot and Noailles describe these operators in [DAN94]. We can define for each aircraft its local fitness:

$$F_i = \sum_{j=1}^{n} (F_{i,j})$$

where $F_{i,j}$ is the matrix described in section 3.3.

The crossover operator is described on the left part of figure 10. After choosing 2 parents A and B, we compare the local fitnesses of their aircraft (These fitnesses are named A_i and B_i on figure 10). For aircraft i, if $A_i \leq B_i - \Delta$ (Δ is a value that modulates the operator's determinism) the 2 children inherit aircraft i of father A. If $B_i \leq A_i - \Delta$, the 2 children inherit aircraft i of father B. If none of these two conditions are checked, aircraft i of children 1 and 2 are two random combination of aircraft i of the two parents.

The mutation operator is described on the right part of figure 10. After choosing a chromosome, an aircraft is mutated (on figure 10 aircraft 4 is chosen). Aircraft having a high fitness (more than ϵ where ϵ modulates the operator's determinism) can be chosen only if every aircraft has a high fitness.

These operators have the great advantage to be rather deterministic at the beginning of the optimization so that a solution without conflict can be very quickly found. When such solutions become sufficiently numerous, these operators are less deterministic and other parts of the research space can be explored.

Two parameters (Δ and ϵ) where introduced and influence the determinism of the optimization. Because of the determinism of these operators, using sharing becomes essential.

3.4.2 Sharing

We have already seen that the problem is very combinatorial and may have many different optimal or nearly optimal solutions. In order to find most of these solutions[6] and to avoid getting trapped in local optima, the sharing process introduced by Yin and Germay [YG93] is used. This sharing process has the great advantage to grow in $n\log(n)$ (instead of n^2 for classical sharing) if n is the size of the population.

A sharing process requires introducing a distance between two chromosomes. Defining a distance between two trajectories is not very simple. In this study, we define five trajectory types:

- the aircraft is not deviated
- the aircraft is deviated on the right for a turning point
- the aircraft is deviated on the left for a turning point
- the aircraft is deviated on the right for an offset
- the aircraft is deviated on the left for an offset

A discrete distance is defined as follow: two chromosomes belong to the same class if each of their aircraft follow the same trajectory types. If not, they belong to different classes. The fitness value of each chromosome is then divided by the number of elements in its class. Results show that sharing was very useful for combinatorial problems.

Simulated annealing and sharing process have really improved convergence of GAs and were definitely adopted.

3.4.3 Improving the result

Genetic algorithms are very efficient to solve global combinatorial optimization problem but are not very efficient to solve local search with a good precision. Consequently, it is very efficient, at the last generation of the genetic algorithm, to use a local optimization method to improve the best solution of each chromosome class defined above. The local method adopted in this study is very simple: we apply a hill-climbing algorithm to the best chromosomes at the end of the GA run.

[6] Finding several solutions is very interesting because the controller can make a choice and negotiate it with pilots.

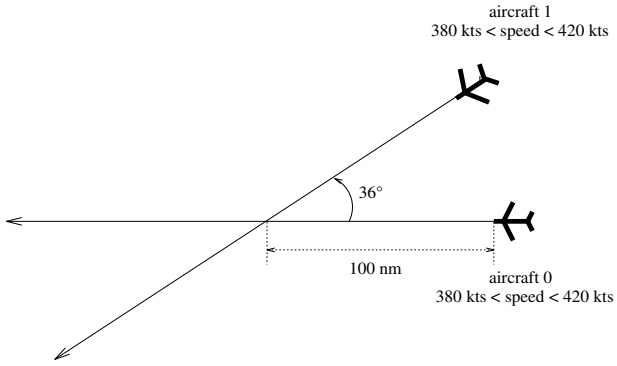

Figure 11: 2 aircraft conflict

3.4.4 Genetic algorithms versus simulated annealing

When dealing with stochastic optimizations, it is always possible to try to use a pure simulated annealing optimization technique. Experiments were done using ASA [Ing89, Ing93, Ing95]. But in this case (which is a good example to the contrary of [IR92]), ASA can not use adapted crossover operators, and results with pure stochastic operators were very poor.

4 Results

We present here three examples to justify the model and illustrate the performance of the algorithm.

4.1 Model Justification

In this first application, we consider the conflict between two aircraft described on figure 11. The anticipation is set to 2 minutes, aircraft speed (400 knots) are known with an error of 5 per cent and trajectory is forecast for 20 minutes. In this case, if aircraft 0 real speed is 380 knots and aircraft 1 real speed is 420 knots, because of uncertainty, at last no conflict occurs (the standard separation is 4 nm). Figures 12,13,14 show the result of the genetic algorithm at time $t = 2$ minutes, 4 minutes, 6 minutes, 8 minutes, 10 minutes and 12 minutes. The thick line describes the next two minutes trajectory that can not be changed, the thin line represents the optimal trajectory computed by the genetic algorithm[7]. Because of uncertainties, the initial optimized trajectory is not very good. As times goes on, aircraft are closer to the conflict point and uncertainty decreases so the optimized trajectories give smaller deviations. Finally, at time $t = 12$, the conflict disappears. This example demonstrates the interest of the model because it allows uncertainties and delays

[7]The parameters of the genetic algorithm were: number of generations: 20, population elements: 50, percentage of crossover: 60, percentage of mutation: 15, simulated annealing for crossover: yes, sharing: yes, hill-climbing method: yes.

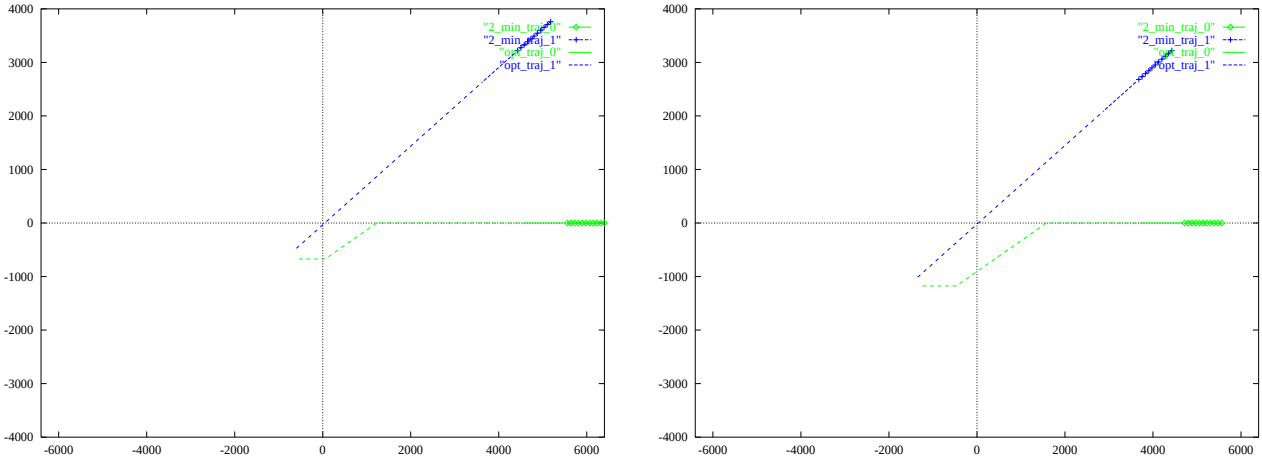

Figure 12: t=2 minutes and t=4 minutes

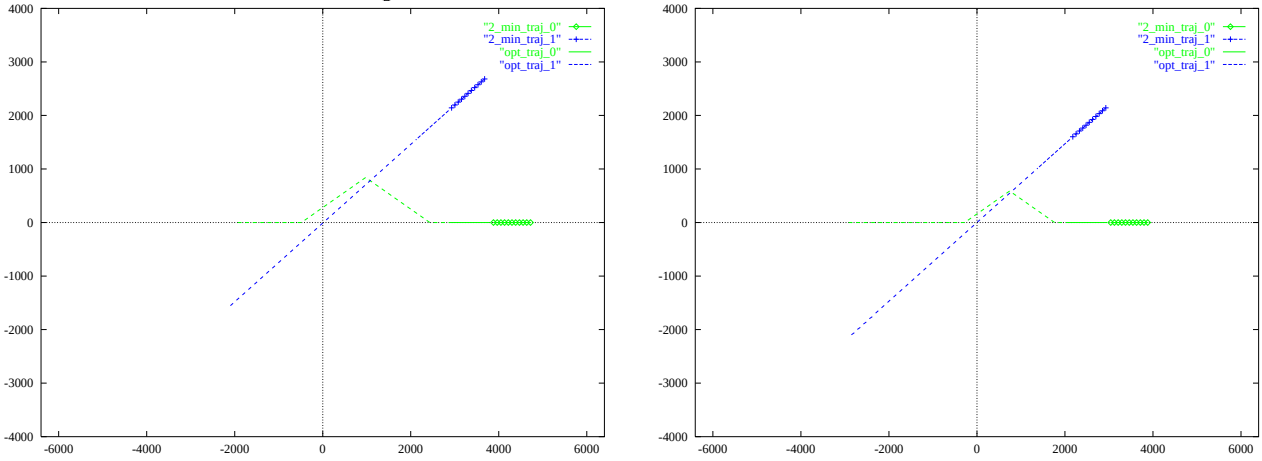

Figure 13: t=6 minutes and t=8 minutes

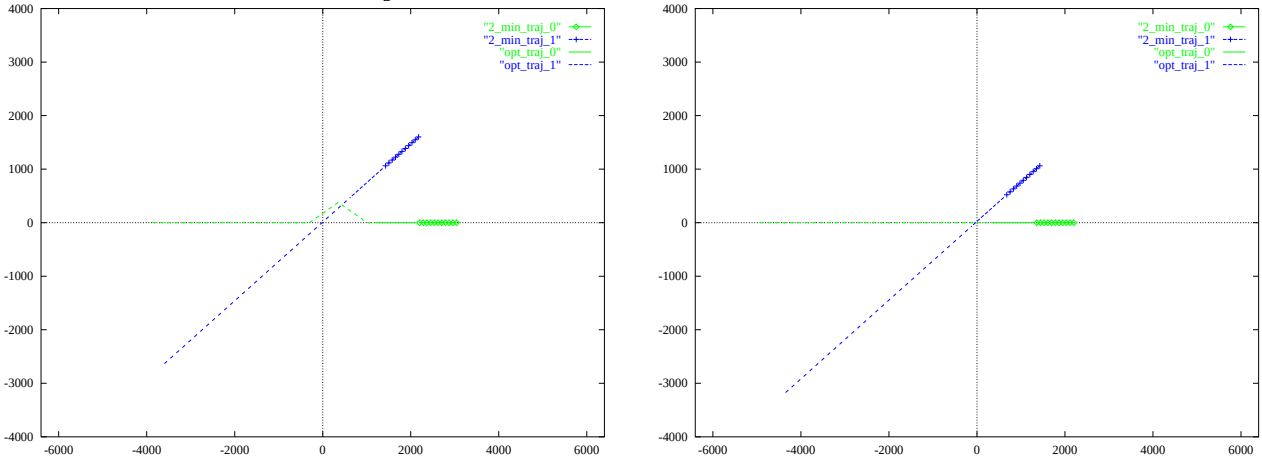

Figure 14: t=10 minutes and t=12 minutes

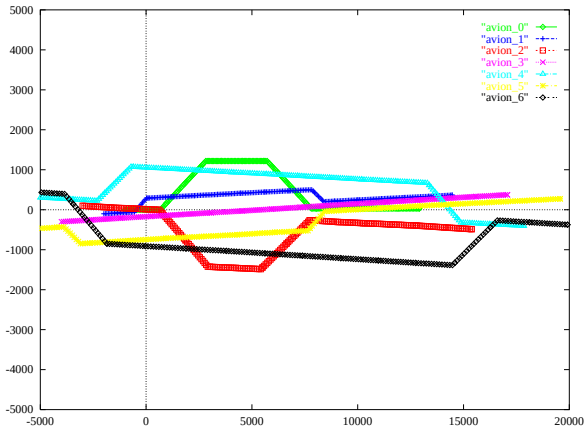

Figure 15: 7 aircraft conflict (aircraft come from the left)

maneuvers as long as possible. Thus, the genetic algorithm gives an optimal command for the next 2 minutes and the optimal trajectory associated.

4.2 Long lasting conflicts

In this test, 7 aircraft are coming from the west and going to the east with different speeds (the furthest has the greatest speed and the nearest the lowest speed). The aircraft are not exactly following the same routes. Figure 15 gives the trajectories of the 7 aircraft with 5 per cent error on speed, 5 minutes of anticipation and 20 minutes trajectory forecast. The size of the population of the genetic algorithm was doubled.

This example was done to show that our model allowed long lasting deviations when they can not be avoided. Here, aircraft 6 is deviated for more than 45 minutes. This example shows the importance of the offset model to solve long lasting conflicts. Solving this problem with turning points would have induced much larger delays and very big deviations (in the example, aircraft 6 would have been deviated several hundreds of nautical miles away from its trajectory).

4.3 Very difficult problems

To push the system to its limits, we decided to test it on two very difficult conflicts involving 20 aircraft.

The first one (left part of figure 16) deals with 20 aircraft on a circle. It has very few solutions because all the aircraft are conflicting with each other. Only solutions where all the aircraft (except perhaps one) turn in the same direction are conflict free. Right part of figure 16 gives the result of the optimization with a speed error of 3 per cent, a 5 minutes anticipation time and a 20 minutes trajectory forecast[8]. The minimum distance between

[8]The parameters of the genetic algorithm were: (number of generations: 20, population elements: 100, percentage of crossover: 60, percentage of mutation: 15, simulated annealing for crossover: yes, sharing:

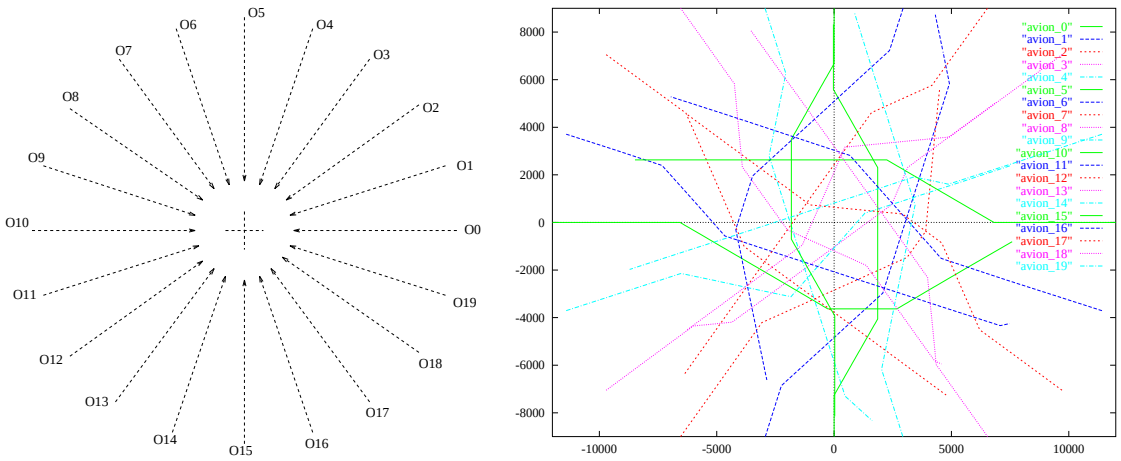

Figure 16: 20 aircraft conflict (aircraft on a circle)

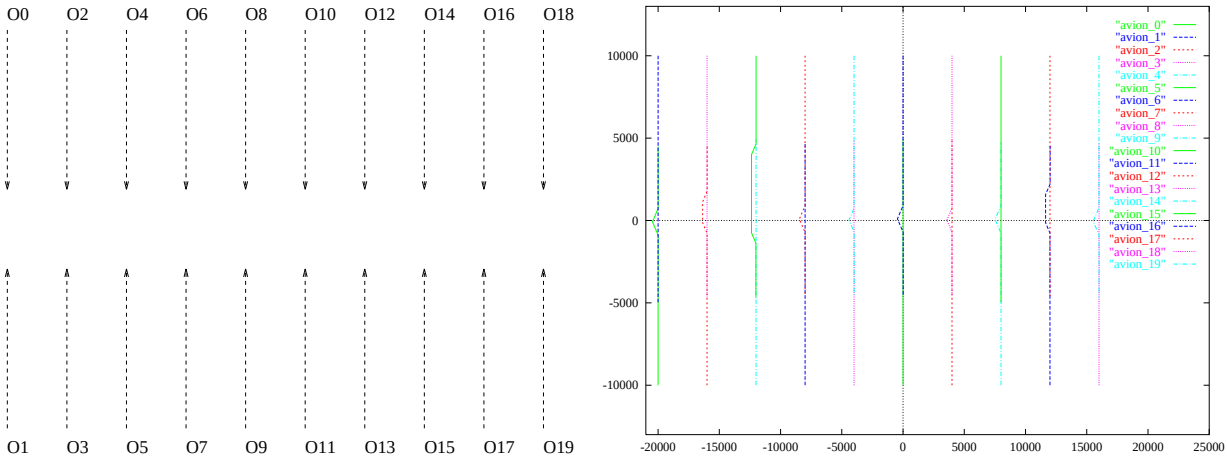

Figure 17: 20 aircraft conflict (10 different conflicts)

aircraft is 5.6 nm (the standard separation was set to 4 nm).

The second conflict (figure 17) deals with 10 independent conflicts between two aircraft. In this example there are many available solutions which can easily be found by a human being because of the symmetries of the problem, but for the Genetic Algorithm, the complexity remains. It is then an excellent validation of the problem solver: right part of figure 17 gives the result of the optimization with the same parameters as previously, and the solution found is optimal. The minimum distance between aircraft is 4.1 nm.

These examples also show that using adapted operators and sharing is mandatory with very difficult problems. These tests were solved with 4 different configurations of the GA:

- no sharing process and adapted operators (described previously).
- a sharing process and adapted operators.

yes, hill-climbing method: yes).

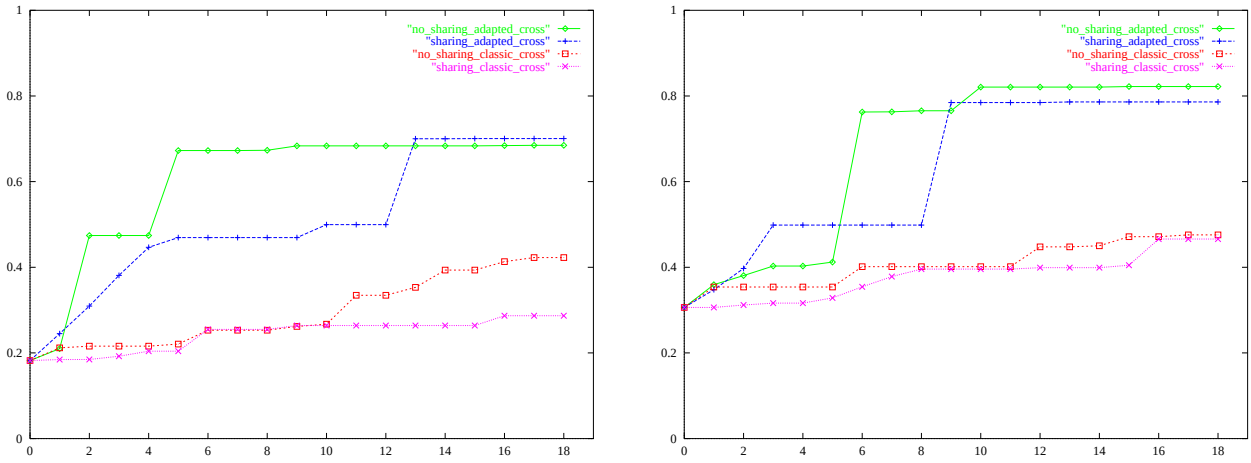

Figure 18: Best fitnesses for two different 20 aircraft conflict

- no sharing process and classical operators.
- a sharing process and classical operators.

For these 4 tests, we have measured on each example the value of the best fitness on the 20 generations of the genetic algorithm (figure 18). First, we can check that adapted operators are very efficient because with classical operators, no conflict free solution is found before generation 20 (tests have shown that the first conflict free solution is found after generation 200) whereas with adapted operators a conflict free trajectory is always found before generation 10 (a solution is conflict free if its fitness is more than 0.5). Figure 18 shows also that the sharing process lengthens the growth of the best fitness. However, the final best fitness is as good with a sharing process as it is without. It is even better with a sharing process when the conflict is very difficult to solve. The sharing process has the great advantage to help the genetic algorithm avoiding local minima and to give different conflict free solutions.

5 The ATC testbench

5.1 General architecture of the test bench

The above tests shows that the problem solver is very efficient on artificial tests. Then we had to test it in a "real" environment.

Therefore we built a complete testbench, using real traffic data as inputs. The architecture of the testbench (figure 19) was the following:

- one process (called P1) is the traffic simulator. It uses unregulated flight plans as inputs. Aircraft depart from airports, climb to their flight level and then descend to their destination. The flight model is a simple one (figure 19): its inputs are the

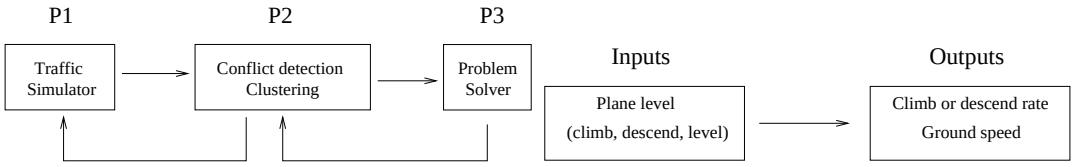

Figure 19: General architecture and flight model

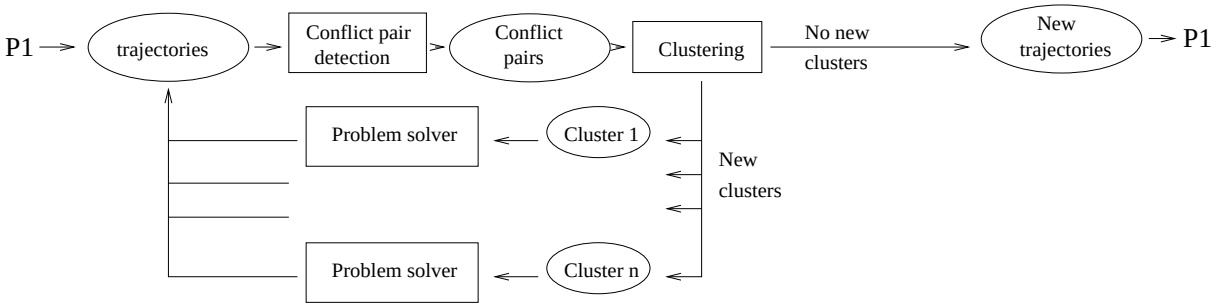

Figure 20: Detailed architecture of the testbench

aircraft level and whether the plane is climbing, descending or levelled. Then, the model gives a ground speed and a climb or descend rate. There is no way to change the climb or descend rate, nor to modify the speed.

- the second process (P2) is in charge of conflict pair detection, clustering of pairs, and verification of new trajectories built by the solvers.
- the third process (P3) is the problem solver.

The system architecture is detailed in figure 20. Process P1 sends current plane positions and flight plans to process P2. Then P2 builds trajectories forecast for the next 20 minutes, and does conflict detection by pairs. The conflict detection model has been described in figure 2; P2 takes into account speed uncertainties as described in part 2.1.1. After pair detection, P2 does a clustering which is a transitive closing on all pairs. Each equivalence class for the relation "is in conflict with", is a cluster.

Then, each cluster is solved in parallel by the problem solver, as each cluster has independent aircraft. The problem solver sends back to P2 modified trajectories. Then P2 once again runs a conflict detection process to check that modified trajectories for aircraft in one cluster does not interfere with aircraft in an other cluster. If no interference is found, resolution orders for the next five minutes are sent to P1. If there are interferences, then the two (or more) clusters interfering are joined and the problem solver is used again on that (these) cluster(s). The process is iterated until no interferences between clusters is detected. The process will always converge: in the worst case, P3 will have to solve a very large cluster including all aircraft that are in conflicts in the next 20 minutes. However, this technique is usually efficient as a very large number of clusters can be solved very quickly

in parallel[9].

5.2 A complete test

Testing of the problem solver is still in progress, but some tests have been already completed [Cha95]. We describe here an experiment ran with flight plans of the 12th of November 1992. There were 4835 aircraft over France. Uncertainties on climbing rate and ground speed were set to 1%, and standard separations were set to 6 Nm and 1000 feet. The experiment was run under the Free Route hypothesis (aircraft were allowed to go directly to their destination). We only detect and solve conflicts above 6000 feet, as we are only interested in En Route conflicts.

When running this one day test with a very basic conflict detection algorithm (only actual conflicts are detected, there is no uncertainty on speed) and with no conflict resolution, 665 conflicts were detected.

When running the complete test bench, the P2 process detected 4408 pairs of aircraft in conflict. Of these 4408 pairs only 908 where different conflicts. This is because P2 is repeated many times (every 5 minutes, and sometimes twice or more in one step because of cluster union). The problem solver had to solve 1888 clusters. 1541 were completely solved. 260 were "almost" solved, which means conflicts not solved at first time finally did not occur because of uncertainties on plane speeds. This lets 80 conflicts not solved. On these 80 conflicts, many of them were different instances of the same one. When keeping only different conflicts, we only had 23 unsolved conflicts. All were conflicts due to aircraft landing, taking off or arriving in the French airspace at the same time and the same place: this is because we use unregulated flight plans. Cutting every conflict under 6000 feet is not enough to suppress all these problems, as aircraft arriving from other airspace, aircraft departing from the same airport and following the same route or aircraft landing at the same airport at the same time generate conflicts that can not be solved.

Results are very interesting, especially if we take into account the fact that the problem solver can only give orders in the horizontal plane.

5.3 Weaknesses of the problem solver

There are still lots of problems to solve:

- the main problem is certainly trajectory forecast. The system is highly sensitive to errors on aircraft speed. In the previous example, speed uncertainties were set to 1%. Other experiments were done with much larger speed uncertainties (up to 7% on ground speed and 20% on climbing and descending rates). There, the system almost saturates: there is a very large number of big clusters, and without resolution orders in the vertical plane, solving conflicts becomes very difficult. Trajectory forecast is definitely a serious issue for all systems doing either automatic resolution or controller

[9] Technicallyspeaking, we use a network of workstations linked by an Ethernet network and we use PVM [GBD+94] to pass messages between applications.

assistance: no controller would accept to work with a system detecting conflicts that never occur, or not detecting conflicts that will occur. The only system to address correctly this issue seems to be ARC-2000 [K+89, FMT93]: as it relies completely on the FMS-4D on board, trajectory forecast is supposed to be almost perfect. But its hypotheses are very strong. With GPS report in a few years, the problem of radar precision will be solved. However, for ground based systems, there will still a lot of work to do. The tabulated model used in our system is definitely too poor, but mass-energy models need data that are not yet available (such as plane mass before departing, or company customs for Mach point setting, etc.) Other more elaborated models describing the complete flight equations set for each plane need even more data that aircraft manufacturers are very reluctant to give. Moreover, wind models are still quite inaccurate. According to us, an adaptive system, either based on symbolic learning or neural network, should be able to adapt dynamically its parameters according to the "standard" plane model and past aircraft positions. Work is currently in progress on that subject.

- Our system needs to be extended to resolution in the z-plane. This is not a serious problem, and will be done in a few months. We do not plan to solve conflicts by speed modifications. Theoretical study shows that optimal En Route conflict resolution by speed modifications would require large anticipation time[10], which is quite unrealistic due to plane speed uncertainties.

- A complete proof of the system is impossible: it is clear that if there are "enough" aircraft, any system would become overloaded and unable to solve conflicts, just because there will not be enough room to move aircraft. On the opposite, with no aircraft, any system can solve every conflict. Proving an ATC system is in no way possible. Anyway, the current ATC system is not proved either, and human beings will always be prone to fail, and will probably be in the control loop for quite a long time.

  This system must be looked upon as a filter in the complete ATC system. Before solving conflicts, there will always be the need for regulations and planification, and in case of unexpected failure of the problem solver, there will always be the need for systems such as the ACAS system to ensure last minute separation.

Conclusion

The model introduced in this paper is a first step in the resolution of real conflicts with real aircraft. The goal of this work was to demonstrate that a mathematical approach, a mathematical modelling and the use of state of the art global optimizations techniques could be a good way to build a problem solver and a complete model for conflict detection

[10] Anticipation time depends on different parameters such as angle of convergence, speed margins for each plane, standard separation etc. More details can be found in [DA95].

and resolution. Even if there are still problems to solve, the results of the test bench are definitely promising.

This work is sponsored since 1992 by the CENA which is the institute in charge of studies for improving the French air traffic control systems.

References

[AGS93] Jean-Marc Alliot, Hervé Gruber, and Marc Schoenauer. Using genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE, 1993.

[AK89] Emile Aarts and Jan Korst. *Simulated annealing and Boltzmann machines*. Wiley and sons, 1989. ISBN: 0-471-92146-7.

[AS92] Jean-Marc Alliot and Thomas Schiex. *Intelligence Artificielle et Informatique Théorique*. Cepadues, 1992. ISBN: 2-85428-324-4.

[BH75] Bryson and Ho. *Applied Optimal Control*. Hemisphere Publishing Corporation, New York, 1975.

[CGT92] A.R. Conn, Nick Gould, and Ph. L. Toint. A comprehensive description of LANCELOT. Technical report, IBM T.J. Watson research center, 1992. Report 91/10.

[Cha95] Olivier Chansou. Résolution automatisée de conflits en route. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1995.

[DA95] Nicolas Durand and Jean-Marc Alliot. Régulation en vitesse : etude théorique. Technical report, CENA/ENAC, 1995.

[DAAS94] Nicolas Durand, Nicolas Alech, Jean-Marc Alliot, and Marc Schoenauer. Genetic algorithms for optimal air traffic conflict resolution. In *Proceedings of the Second Singapore Conference on Intelligent Systems*. SPICIS, 1994.

[DAM93] Patrick Dujardin, Jean-Marc Alliot, and Paul-Henri Mourlon. Different paths to automation. In *IFAC'93*, 1993.

[DAN94] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Algorithmes genetiques : un croisement pour les problemes partiellement separables. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 1994.

[DAN96] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Submitted to ACM/SAC'96*. ACM, 1996.

[DASF94a] Daniel Delahaye, Jean-Marc Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for air traffic. In *Proceedings of the Conference on Artificial Intelligence Application*. CAIA, 1994.

[DASF94b] Daniel Delahaye, Jean-Marc Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for partitioning airspace. In *Proceedings of the Tenth Conference on Artificial Intelligence Application*. IEEE, 1994.

[Dur94] Nicolas Durand. Conflict free trajectory modelling for en route control. Technical report, ENAC/CENA, January 1994.

[E.K82] E.Kreindler. Additional necessary conditions for optimal control with state-variable inequality constaints. *Journal of Optimization theory and applications*, 38(2):241–250, october 1982.

[FMT93] Xavier Fron, Bernard Maudry, and Jean-Claude Tumelin. Arc 2000 : Automatic radar control. Technical report, Eurocontrol, 1993.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. Pvm 3 user's guide and reference manual. Technical report, Oak Ridge National Laboratory, 1994.

[Gol89] David Goldberg. *Genetic Algorithms.* Addison Wesley, 1989. ISBN: 0-201-15767-5.

[Hol75] J.H Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan press, 1975.

[Ing89] Lester Ingber. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12:967–973, 1989.

[Ing93] Lester Ingber. Simulated annealing: practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[Ing95] Lester Ingber. Adaptive simulated re-annealing (asa): lessons learned. *Control and cybernetics*, 1995.

[IR92] Lester Ingber and Bruce Rosen. Genetic algorithms and very fast simulated re-annealing: a comparison. *Mathematical and computer modeling*, 16(11):87–100, 1992.

[JWS63] A.E.Bryson Jr, W.F.Denham, and S.E.Dreyfus. Optimal programming problems with inequality constraints: Necessary conditions for extremal solutions. *AIAA Journal*, 1:2544–2550, november 1963.

[K+89] Fred Krella et al. Arc 2000 scenario (version 4.3). Technical report, Eurocontrol, April 1989.

[MÓ94] Frédéric Médioni. Algorithmes génétiques et programmation linéaire appliqués a la résolution de conflits aériens. Master's thesis, Ecole Nationale de l'Aviation Civile (ENAC), 1994.

[MDA94] F. Medioni, Nicolas Durand, and J.M. Alliot. Algorithmes génétiques et programmation linéaire appliqués a la résolution de conflits aériens. In *Proceedings of the Journees Evolution Artificielle Francophones.* EAF, 1994.

[MG92] Samir W. Mahfoud and David E. Goldberg. Parallel recombinative simulated annealing: a genetic algorithm. IlliGAL Report 92002, University of Illinois at Urbana-Champaign, 104 South Mathews Avenue Urbana IL 61801, April 1992.

[Mic92] Zbigniew Michalewicz. *Genetic algorithms+data structures=evolution programs.* Springer-Verlag, 1992. ISBN: 0-387-55387-.

[NFC+83] W.P. Niedringhaus, I. Frolow, J.C. Corbin, A.H. Gisch, N.J. Taber, and F.H. Leiber. Automated En Route Air Traffic Control Algorithmic Specifications: Flight Plan Conflict Probe. Technical report, FAA, 1983. DOT/FAA/ES-83/6.

[Nie89a] W.P. Niedringhaus. Automated planning function for AERA3: Manoeuver Option Manager. Technical report, FAA, 1989. DOT/FAA/DS-89/21.

[Nie89b] W.P. Niedringhaus. A mathematical formulation for planning automated aircraft separation for AERA3. Technical report, FAA, 1989. DOT/FAA/DS-89/20.

[vKHHK95] C.H.M. van Kemenade, C.F.W. Hendriks, H.H. Hesselink, and J.N. Kok. Evolutionnary computation in air traffic control planning. In *Proceedings of the Sixth International Conference on Genetic Algorithm.* ICGA, 1995.

[YG93] Xiaodong Yin and Noel Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In C.R. Reeves R.F.Albrecht and N.C. Steele, editors, *In proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference, Insbruck Austria.* Springer-Verlag, 1993.

[Zeg94] Karim Zeghal. *Vers une thèorie de la coordination d'actions. Application à la navigation aérienne.* PhD thesis, Université Paris VI, 1994.