??, ??, 1–10 (??)

Neural nets trained by genetic algorithms for collision avoidance

NICOLAS DURAND, JEAN-MARC ALLIOT

durand@recherche.enac.fr, alliot@recherche.enac.fr

Centre d'Etudes de la Navigation Arienne

FRÉDÉRIC MÉDIONI

Centre de Mathématiques Appliquées de l'Ecole Polytechnique

Received ??; Revised ??

Abstract. As air traffic keeps increasing, many research programs focus on collision avoidance techniques. For short or medium term avoidance, new headings have to be computed almost on the spot, and feed forward neural nets are susceptible to find solutions in a much shorter amount of time than classical avoidance algorithms (A^*, stochastic optimization, etc.) In this article, we show that a neural network can be built with unsupervised learning to compute nearly optimal trajectories to solve two aircraft conflicts with the highest reliability, while computing headings in a few milliseconds.

Keywords: Air Traffic Control, Collision Avoidance, Neural Networks, Genetic Algorithms

1. Air traffic control (ATC) and collision avoidance

As air traffic keeps increasing, the ATC system overload becomes a serious concern. For the last twenty years different approaches have been tried and many solutions proposed, originating with the AERA-II and AERA-III projects [17, 15, 14]. In a few words, all theses solutions are between the two following extreme positions:

On the one hand, we can imagine an ATC system where each aircraft would follow its planned trajectory with a perfect accuracy. With such a system, no reactive system would be needed as conflicts [1] could be solved before aircraft take off. This solution is close to the ARC-2000 hypothesis, which has been investigated by the Eurocontrol Experimental Center [12].

On the other hand, there could be an ATC system where trajectories are not planned. Each aircraft flies its own way, and all collisions are to be avoided by reactive systems. Each aircraft would be in charge of its own safety. This could be called a completely free flight system. The free flight hypothesis is currently seriously considered for all aircraft flying "high enough".

Of course, no ATC system will ever totally rely on only one of these two hypothesis. It is quite easy to understand why. A completely planned ATC is impossible, as no one can guarantee that every trajectory would be perfectly followed; there are many parameters that can not be perfectly forecasted such as meteorological conditions (storms, winds, etc.), breakdowns in aircraft motor, flaps or other problems (closing of landing runaway on airports, etc.). On the other hand, a completely reactive system looks difficult to handle; it would only perform local optimizations for trajectories. Moreover, in the vicinity of departing and landing areas, the density of aircraft is

so high that trajectories generated by this system could soon become Brownian movements.

An ATC system can be represented by a set of filters, or shells. A classical view of the shells in an ATC system could be:

1. As many aircraft are simultaneously present in the sky, a single controller is not able to manage all of them. So, airspace is divided into sectors, each of them being assigned to a controller. Airspace design aims at designing the air network and the associated sectoring.
2. Air Traffic Flow Management (ATFM) (strategic planning, a few hours ahead): With the increasing traffic, many pilots choose the same routes, generating many conflicts over the beacons inducing overloaded sectors. Traffic assignment aims at changing aircraft routes to reduce sector congestion, conflicts, and coordinations.
3. The coordination planning (a few minutes before entering in the sector) guarantees that each new aircraft entering a control sector does not overload the sector.
4. Tactical control in ATC centers (up to 20 minutes ahead): At this level, controllers solve conflicts between aircraft.
5. Collision avoidance systems (a few minutes before collision): This shell is activated only when the previous one has failed. It is not supposed to be activated in normal situations.

Each level has to limit and organize the traffic it passes to the next level, so that this one will never be overloaded.

In this paper, we present a problem solver that can handle the collision avoidance problem (level 5 filter) with reactive techniques. This problem solver is based on a neural network, which is built by a genetic algorithm. Building neural networks with GA has already been done. Application quite similar can be found in the literature such as car parking [16], or chromatography [8].

2. Existing reactive techniques

The most well known concept on reactive collision avoidance is certainly the ACAS^2 concept. It is already implemented in its two first versions (TCAS-I and TCAS-II) and only implements manoeuvre in the vertical plane (extension to the horizontal plane [1] were inconclusive). It is a very short term collision avoidance system (less than 60 seconds). It should only be thought as the last security filter of an ATC system. Using TCAS to control aircraft would probably end in serious problems. The TCAS algorithm is based on the application of a sequence of filtering rules, which give the pilot a resolution advice.

Another simple technique has been investigated by [10]. The idea is to consider each aircraft as positive electric charges, while the destination of the aircraft is a negative charge. Each aircraft creates a repulsive force proportional to the inverse of the square of the distance, while the destination behaves like an attractor. This technique has a serious drawback. Symmetries can not be broken. This problem was solved by [20, 19, 18]. The general idea is to add non symmetrical force: a force which has the direction of the repulsive force +90 degrees, and a module which is a small fraction of the module of the repulsive force is added to the repulsive force. This system solves the symmetrical problem. However, there are still some drawbacks: the different parameters of the attractive and repulsive forces are arbitrarily set, and it is unclear to define how to find optimal values. Moreover, the shape itself of the forces is also arbitrarily set. But the main problem of this system is that it forces aircraft to modify their headings, but also their speeds. Unfortunately, the range of available speeds is very limited for aircraft flying at their requested flight level. Moreover, it is technically very difficult to change aircraft speed with a continuous command because it can damage aircraft engines.

Our system only allows heading modification and solves very complex two aircraft conflict, with almost optimal trajectories. Moreover, the system is very fast, as soon as the neural network has been built.

3. Mathematical complexity

If we consider the two aircraft problem, it can be proved, using the residue theorem [6], 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 con-

flict 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. 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.

In each connected component, Optimal Control theory can be used to optimize aircraft trajectories. However, for the collision avoidance problem, an improved version of the Pontriaguine maximum principle is required to take the separation constraint into account. Durand detailed in [6] the conflict resolution problem using the Optimal Command theory [3, 11]. This led to the following conclusions:

1. if aircraft speed is not constrained, an analytical solution can be found (however, this hypothesis on aircraft speed is not realistic).
2. if aircraft speed is constrained, at the optimum, as long as the separation constraint is not saturated, aircraft fly in straight line. When saturating the constraint, aircraft start turning, and as soon as the separation constraint is freed, aircraft fly straight again.
3. when moving only one aircraft, trajectories are also regular and do not include discontinuous points. Moreover, the length of the trajectory increases when the angle of incidence between the two aircraft decreases, the speed ratio gets close to 1, or aircraft are closer to the conflict point when the resolution starts.

For a conflict involving 2 aircraft, local optimization tools such as LANCELOT[3] [4] can solve the collision avoidance problem. However LANCELOT is quite slow and can not be used for a real time application. For more than 2 aircraft, LANCELOT can not be used and other techniques have to be investigated [7, 5].

4. Modeling the problem

The problem we want to solve is the following. An aircraft flying at a constant speed detects another aircraft flying at the same altitude (more or less 1000 feet) in a 20 nautical miles diameter disk. We want to build a neural network that modifies the heading of this aircraft when there is a conflict. The heading must not be changed of more than 45 degrees per 15 seconds for operational reasons. The other aircraft is supposed to have the same embarked system so that it also detects the first aircraft and reacts using the same neural network with different inputs.

The system uses an on board radar to detect other aircraft. Consequently, all the inputs of the neural network must be given by the on board radar information.

The horizontal separation standard is noted n_h and is equal to 4 NM.

5. Using a neural network

Conflict avoidance takes place on a time period of length t_f. The position of an aircraft at time $t = 0$ is called its *initial position*, its position at time $t = t_f$ is called its *final position*, or its *destination*. In our problem, it seems clear that if no conflict occurs, no neural network is needed to solve it. Consequently, at each time step, we first check if both aircraft can connect their destination without changing their headings and without generating conflicts. In that case, we do not modify aircraft headings.

5.1. *The inputs*

9 inputs are used by the neural network (see figure 1). An important data to define these inputs is the heading an aircraft should follow to go directly from its current position to its destination. This heading is called the *direct heading*. Aircraft are noted a_i, for $i \in \{1, 2\}$. The speed of aircraft a_i is noted v_i, its heading is noted h_i, its direct heading is noted h_i^d. The difference between these two heading is $\alpha_i = h_i^d - h_i$. The relative speed of aircraft a_i with respect to aircraft a_j is noted $v_{i,j}$. We describe the inputs used by the neural network that modifies the trajectory of aircraft a_1:

- $\sin\alpha_1$ and $\cos\alpha_1$; we use both $\sin\alpha_1$ and $\cos\alpha_1$ to represent α_1, to maintain continuity of the function when planes cross the 360 degrees boundary

- $\frac{v_2 - v_1}{v_1}$
- $\frac{1}{\delta_d}$, with $\delta_d = \max(60; \|d - n_h\|, 1)$, where d is the distance between the two aircraft and is expressed, as well as n_h, in nautical miles.
- $\frac{v_{2,1}}{v_{max} - v_{min}}$ where v_{max} and v_{min} are the bounds of the possible values of the speed of the aircraft.
- $\sin\gamma_2$ and $\cos\gamma_2$, where $\gamma_2 = h_2 - h_1^d$.
- $\frac{\beta}{360}$, where β is the converging angle of the trajectories (in degrees).
- A bias set to 1.

5.2. *The neural network structure*

The neural network structure used is as simple as possible. A 3 layers network is used (see figure 2) and returns a heading change of 45 degrees maximum (for a time step of 15 seconds). The activation function used is the following:

$$act(s) = \frac{1}{1 + e^{-s}}$$

The first layer has the 8 inputs described above plus the bias. The second layer holds 25 units, while the third layer holds the output unit[4].

5.3. *Learning the neural network weights*

Classical back propagation of gradient can not be used in our case because conflict free trajectories are not known in every configuration. They could be calculated for conflicts involving $n = 2$ aircraft, but the problem is not solvable for $n > 2$. As we plan to extend our system to more than two aircraft, we decided to use unsupervised learning with GA. However, we compare the results of our network with optimal trajectories computed by LANCELOT to validate our hypothesis.

6. Genetic algorithms

Figure 3 describes the main steps[5] of GAs that are used in this paper: first the problem is *coded* and 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.

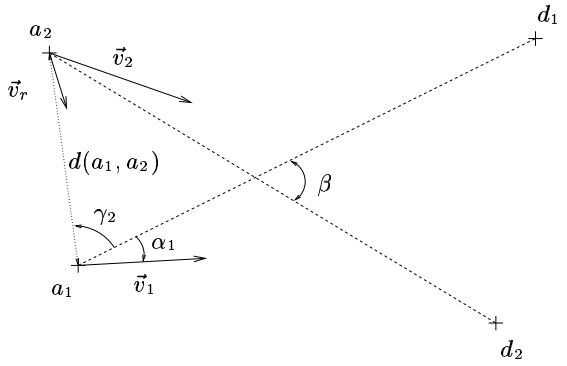

Fig. 1. The neural network inputs of aircraft 1.

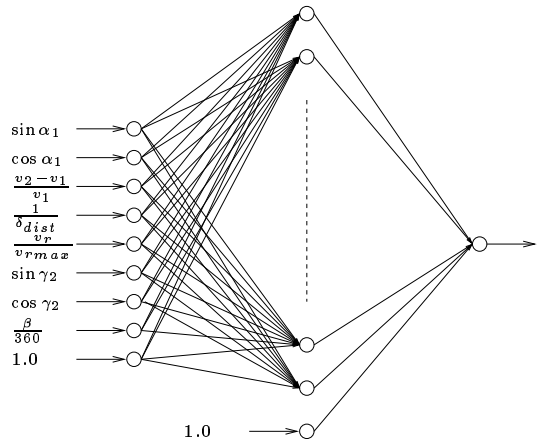

Fig. 2. The neural network structure.

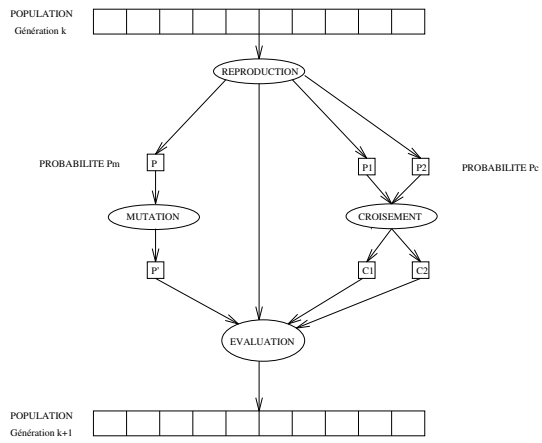

Fig. 3. GA principle

6.1. Coding the problem

Here, each neural network is coded by a matrix of real numbers that contains the weights of the neural network.

6.2. Computing the fitness

One of the main issues is to know how to compute the *fitness* of a chromosome. The constrained problem to solve takes the following criteria into account:

- Aircraft trajectories must be conflict free.
- Delay due to deviation must be as low as possible.
- The fitness of a network which leads to trajectories that do not respect the separation constraints should always be lower than the fitness of a network that leads to trajectories that respect these constraints.

To compute the fitness, a panel of N different conflict configurations[6] is created randomly (*cf* section 6.6).

For these N configurations, we define C as the total number of time steps for which one separation constraint is violated and μ as the quadratic mean of delays. Fitness[7] is defined by:

If $C \neq 0$:

$$f_a = \frac{1}{1+C} \tag{1}$$

If $C = 0$:

$$f_a = 1 + \frac{1}{1+\mu} \tag{2}$$

6.3. Selection

"Stochastic Remainder Without Replacement" [9] is used for selection, along with *ranking*. After the *raw* fitness f_i^r of the n elements of the population is computed, these fitnesses are scaled; the elements of the population are ranked, according to their fitness: the best element gets rank 1, and the worst one gets rank n. The rank of an element is noted r_i. The scaled fitness is defined by: $f_i^s = \frac{n-r_i}{n}$. Each element is reproduced $\lfloor p_i \rfloor$ times in the new population, with $p_i = n \times f_i / \sum_j f_j$. Then we compute $r_i = p_i - \lfloor p_i \rfloor$, and the population is randomly completed by choosing elements with the probability $r_i / \sum_j r_j$. Number of elements is 500.

6.4. Crossover

The arithmetic crossover is used: 2 parents are recombined by choosing randomly $\alpha \in [-0.5, 1.5]$ and creating child 1 (resp child 2) as the barycentre of some randomly chosen weight of $(parent_1, \alpha)$ (resp $(parent_1, 1-\alpha)$) and $(parent_2, 1-\alpha)$ (resp $(parent_2, \alpha)$). Crossover probability is 60%.

6.5. Mutation

The mutation operator adds a Gaussian noise to one of the weights of the neural network. The mutation probability is set to 15%.

6.6. The learning examples

The learning set contains $N = 50$ conflict configurations. These configurations are generated randomly and remain unchanged throughout learning (fixed learning test). The position of an aircraft at time $t = 0$ is its initial position; its position at time $t = t_f$, if it is not deviated, is called its final position. The configurations generated are such that:

- The distance between two aircraft at time $t = 0$ is equal to an *alert distance* noted d_a.
- If aircraft are not deviated, a conflict occurs between $t = 0$ and $t = t_f$, but aircraft are separated again at time $t = t_f$.

Two configurations are considered to be equal if it is possible to get from one to the other through a translation or a rotation (the inputs of the neural network use only relative positions of aircraft). Configurations are generated in order to represent as much as possible all relative positions and all relative headings at the beginning of the conflict. The speed of the aircraft ranges between $v_{min} = 300\ kts$ and $v_{min} = 500\ kts$.

7. Numerical results

7.1. Preliminary results

To evaluate the performance of the neural network, we have tested it on a large number (10000) of non learned conflict configurations. These conflict configurations are generated randomly. The neural network generated conflict free trajectories for 9612 out of 10000 configurations (4 % failure). In most cases, the violation of the separation constraint is not very important: for 152 configurations, the minimal distance between the two aircraft is higher than 3.75 NM, it is between 3 and 3.75 NM for 195 configurations, and never gets below 2 NM. The mean delay of aircraft (on the configurations for which conflict free trajectories are found) is 5.1 seconds. The delay of aircraft is lower than 10 seconds for 7802 configurations, higher than 30 seconds for only 10 configurations, and never higher than 1 minute. On the average, an aircraft is in conflict every 30 minutes. So, the average delay is 0.3%.

7.2. Improving results

The reliability of the neural network that learned on a fixed learning test is quite good (4 % failure), but is not perfect. To improve it we have used a renewed learning set, along with a different way of computing the fitness of a network.

The conflict configurations of the learning set are renewed at each generation of the genetic algorithm. They are replaced by other conflict configurations, randomly generated.

A neural network, if it survives for several generations, has been confronted to different learning sets. These different learning sets are used to compute the network fitness. We modify the definitions of C and μ. We define C', the mean value, on the different learning sets to which the neural network has been confronted, of the total number of time steps on which aircraft are not separated (for the N conflict configurations of each learning set), and μ' the mean value on these different learning sets of the quadratic mean value of the sum of the delays of the two aircraft on the N conflict configurations of each learning set.

We also use a reliability factor, adapted of a concept used to train a program designed to play Othello games [2]. This program was evaluated by counting the number of victories against a reference program. The reliability of a program depends on this number, but also on the number of games already played. A program that won 46 games out of 48 may be more reliable than one that won 6 games out of 6.

Let us suppose that the probability that the program wins a game is p. The probability that it wins m games out of n is then:

$$P(p, m, n) = \binom{n}{m} p^m (1-p)^{(n-m)} \qquad (3)$$

Let us suppose that the program has won m games out of n. It can be then shown that for $p_f \in [0, 1]$, p is higher than a certain value p_{m,n,p_f} with probability p_f, with the following implicit definition of p_{m,n,p_f}:

$$\int_{p_{m,n,p_f}}^{1} P(p, m, n) dp = \frac{p_f}{n+1} \qquad (4)$$

Let n_s be the number of successive learning sets for which the network generated only conflict free trajectories. We define s_r as the reliability factor of the network:

$$s_r = p_{n_s, n_s, p_f}$$

with $p_f = 0.95$. The different values of s_r for the different possible values of n_s (here 1 to 1000, which is the maximal number of generation, and thus of learning sets) are computed once before the genetic algorithm is run.

The reliability factor s_r of a network is used to compute its fitness:

If $C' \neq 0$:

$$f_a = \frac{1000}{1 + C'} \tag{5}$$

If $C' = 0$:

$$f_a = 1000 + s.\frac{1000}{1 + \mu'} \tag{6}$$

7.3. Results with the renewed learning set

Results are excellent. The new network has been tested on 10000 configurations, and generated conflict free trajectories for all of them. In terms of delays, the results are a little less satisfying: the mean delay is then 7.5 seconds (it is 5.1 seconds for the fixed learning set), the delay of aircraft is lower than 10 seconds for 6838 configurations, between 10 and 30 seconds for 3019 configurations, between 30 seconds and 1 minute for 132 configurations. The delays are higher than 1 minute in 18 cases, but never higher than 2 minutes.

So, there is a minimal loss of performance regarding delays, but separation is now enforced. Neural networks learned with renewed learning sets are much better than the ones learned with a fixed set.

7.4. Comparison with LANCELOT

Optimal solutions to the different configurations are calculated using gradient method such as LANCELOT. LANCELOT has the great advantage to find the optimal solution to our problems but requires much more time (one hour on HP720). Controlling aircraft in real time with this technique is not possible. However, it is interesting to compare optimal solutions found by LANCELOT to solutions computed by the neural network.

The configurations used to compare the neural network to optimal solutions are not learned configurations. For each solution, we give the mean lengthening of the trajectories in percentage:

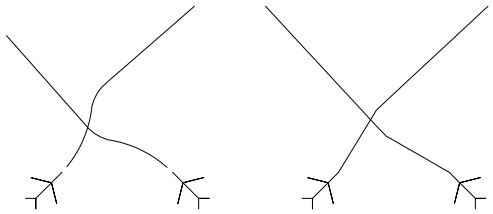

Fig. 4. Neural network solution (left), optimal solution (right).

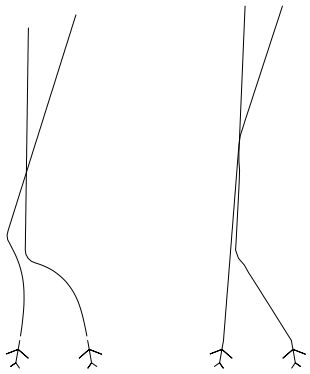

Fig. 5. Neural network solution (left), optimal solution (right).

- Figure 4 gives an example of conflict at 90 degrees in which aircraft have the same speed. Neural network (1.08%) and optimal solution (0.26%) are similar. The NN solution mean lengthening is worse than the optimal solution lengthening.
- Figure 5 gives an example of a 15 degrees conflict where aircraft have the same speed. Such a conflict is particularly difficult to solve. Solutions are different, but for such a difficult conflict, the neural network (2.30%) gives a

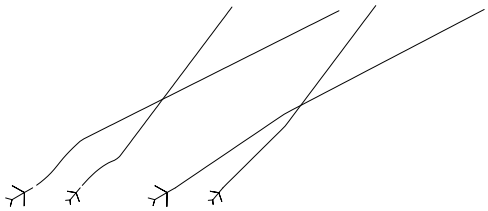

Fig. 6. Neural network solution (left), optimal solution (right).

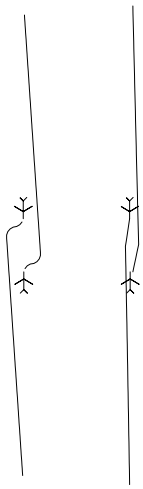

Fig. 7. Neural network solution (left), optimal solution (right).

solution that is robust and quite as good as the optimal solution (2.23%). This conflict is the most difficult conflict to solve (in the 5 examples presented). It is interesting to see that the difference of lengthening is the smallest.

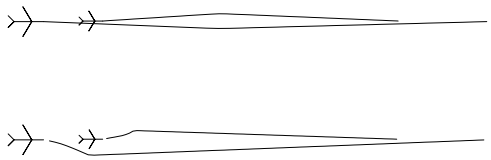

Fig. 8. Neural network solution (down), optimal solution (up).

- Figure 6 gives an example of aircraft at different speeds (400 and 500 knots) with crossing at a small angle (30 degrees). The neural network solution (1.32%) appears very similar to the optimal solution (0.28%) but it is less efficient.
- Figure 7 gives an example of aircraft crossing on the same route. This problem is easy to solve and solutions are similar. The NN solution (1.18%) is robust but worse than the optimal solution (0.25%).
- Figure 8 gives an example of aircraft flying on parallel routes at different speeds. This problem is easy to solve. Solutions are similar. The NN solution (1.02%) is robust but worse than the optimal solution (0.21%).

These 5 examples show that, if solutions are obviously less optimal, the loss of optimality is not significant (the delay induced by the neural network is always less than 4 times the minimal delay found with LANCELOT, which is generally very small). Tests done on non-learned situations gave results as good as tests done on learned configurations.

8. Conflicts involving 3 aircraft

We wanted to test the possibility of extending resolution to more than 2 aircraft. The three follow-

ing techniques to solve conflicts involving 3 aircraft are used:

Closest intruder: a neural network such as described in section 5 is used. At each time step, the inputs are computed by considering only the closest of the two other aircraft. Aircraft take a direct heading towards their final position as soon as *all* aircraft can do it without conflict.

Threatening intruder: at each time step, each aircraft computes its direct trajectory to its destination and finds the closest aircraft that would be in conflict if they all follow direct routes. Inputs of the network are then computed regarding only this aircraft. An aircraft takes a direct heading towards its destination as soon as *it* is not in conflict if all aircraft fly a direct route.

two intruders: a larger neural net is used, which takes 15 inputs. It uses the same first 9 inputs as in section 5, but 6 more inputs are computed regarding the second aircraft in conflict (they are similar to the ones described in section 5). The hidden layer is extended to 30 neurons. An aircraft takes a direct heading towards its final position as soon as *all* aircraft can do it without conflict.

These three techniques have been used, with a fixed and a renewed learning set. With the fixed learning set, results are good, for the three techniques on learned configurations: conflict free trajectories are generated for all learned configurations, delays are reasonable, though more important than for two aircraft (mean delays around 30 seconds for the three techniques, with a light advantage for the closest intruder technique). Statistical results are quite bad on non learned configurations (10 % failure).

Results with the renewed learning set are not totally satisfying either:

Closest intruder: conflict free trajectories are generated for 9983 out of the 10000 non learned configurations (0,2% failure). But delays are large: 53 seconds mean delay, between 3 and 10 minutes for 296 configurations, larger than 10 minutes for 15 configurations.

Threatening intruder: the failure rate is less than 0.4%, but results are better regarding delays: 40 seconds mean delay, delay exceeding 3 minutes for 144 configurations, never larger than 10 minutes.

Two intruders: the failure rate is very low: 0.06%, but delays are even larger: they exceed 3 minutes for 800 configurations, and exceed 10 minutes for 54 configurations.

9. Conclusion

Using a simple neural network to solve a conflict between 2 aircraft gives very good results. The neural network can be easily learned by a genetic algorithm without knowing the optimal solutions. Robustness of the NN can be improved if new conflict configurations are used at each generation of the genetic algorithm.

Extending the problem to conflicts involving more then 2 aircraft is much more difficult. The *closest intruder* and *closest threatening intruder* techniques are advantageous because they can be extended to more than 3 aircraft. But they seem less robust to non learned configurations than the *two intruders* technique. The latter gives good results regarding the robustness to non learned conflicts but delays are quite important. Furthermore, extension to more than 3 aircraft would make the size of the NN increase and the learning more difficult.

These results are not surprising; as many reactive techniques, NN must be considered as an intermediate filter between the TCAS and tactical resolution techniques. As such, they will operate on simple (mainly 2 aircraft), short/medium term conflicts. For such applications, they are an excellent system, as they combine very fast, real time, computation of new headings and a great reliability and efficiency.

Notes

1. 2 aircraft are said to be in conflict if their altitude difference is less than 1000 feet (305 meters) and the horizontal distance between them is less than 8 nautical miles (14800 meters). These two distances are respectively called vertical and horizontal standard separation
2. Airborne Collision Avoidance System

3. Large And Nonlinearly Constrained Extended Lagrangian Optimization Techniques
4. Different number of units were tried. With less than 25 units, results were not satisfactory. With more than 25 units, results show no evidence of improvements, while training times were longer.
5. We use classical Genetic Algorithms and Evolutionary Computation principles such as described in the literature [9, 13].
6. N represents the number of conflict configurations on which *each* element of the population is tested while n represents the number of elements in the population.
7. The GA is not very sensitive to the exact form of the fitness function. The one choosen is both simple and efficient.

References

1. TCAS-III collision avoidance algorithms version 3. Technical report, The MITRE Corporation, November 1990.
2. J.M. Alliot. A genetic algorithm to improve an othello program. In *Artificial Evolution 95*. Springer, 1995.
3. Bryson and Ho. *Applied Optimal Control*. Hemisphere Publishing Corporation, New York, 1975.
4. 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.
5. N. Durand and J.M Alliot. Optimal Resolution of En Route Conflicts. In *1 ST U.S.A/EUROPE ATM R & D Seminar*, Mai 1997.
6. Nicolas Durand. *Optimisation de Trajectoires pour la Résolution de Conflits en Route*. PhD thesis, ENSEEIHT, Institut National Polytechnique de Toulouse, 1996.
7. Nicolas Durand, J. M. Alliot, and Joseph Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.
8. Alessandro Fadda. *Utilisation de techniques neurogenetiques pour la resolution de problemes inverses*. PhD thesis, Ecole Polytechnique de Paris, 1998.
9. David Goldberg. *Genetic Algorithms*. Addison Wesley, 1989. ISBN: 0-201-15767-5.
10. H. Gruber. Comparaison de diverses méthodes d'intelligence artificielle pour la résolution de conflit en contrôle de trafic aérien. Rapport de stage, Centre d'Etudes de la Navigation Aérienne, 1992.
11. R. F. Hartl, S. P. Sethi, and R. G. Vickson. A survey ot the maximum principles for optimal control problems with state constraints. *SIAM Review*, 1995.
12. Fred Krella et al. Arc 2000 scenario (version 4.3). Technical report, Eurocontrol, April 1989.
13. Z. Michalewicz. *Genetic algorithms+data structures=evolution programs*. Springer-Verlag, 1992. ISBN: 0-387-55387-.
14. W.P. Niedringhaus. Automated planning function for AERA3: Manoeuver Option Manager. Technical report, FAA, 1989. DOT/FAA/DS-89/21.
15. W.P. Niedringhaus. A mathematical formulation for planning automated aircraft separation for AERA3. Technical report, FAA, 1989. DOT/FAA/DS-89/20.
16. Marc Schoenauer, Edmund Ronald, and Sylvain Damour. Evolving nets for control. Technical report, Ecole Polytechnique, 1993.
17. E. M. Schuster, F. R. Petroski, R. K. Sciambi, and M. MC Stokrp. AERA 2 functional design and performance description. Technical report, MITRE, September 1983. MtR-83W136.
18. Karim Zeghal. A comparison of different approaches based on force fields for coordination among multiple mobile. In *IEEE International Conference on Intelligent Robotic System (IROS*, Mai 1993.
19. Karim Zeghal. A Reactive Approach for Distributed Air Traffic Control. In *International Conference on Artificial Intelligence & Expert Systems*, Mai 1993.
20. Karim Zeghal. *Vers une théorie de la coordination d'actions, application à la navigation aérienne*. PhD thesis, Universite Paris VI, 1994.

Nicolas Durand graduated from the Ecole Polytechnique de Paris in 1990 and from the Ecole Nationale de l'Aviation Civile (ENAC) in 1992. He has been a design engineer at the Centre d'Etudes de la Navigation Aérienne (CENA) since 1992 and has completed a Ph.D. in computer science on "Computing Optimal Trajectories for Conflict Resolution".

Jean-Marc Alliot graduated from the Ecole Polytechnique de Paris in 1986 and from the Ecole Nationale de l'Aviation Civile (ENAC) in 1990. He also holds a Ph.D. in computer science (1992). He is currently in charge of the global optimization laboratory of CENA and ENAC in Toulouse.

Frédéric Médioni graduated from the Ecole Polytechnique de Paris in 1992 and from the Ecole Nationale de l'Aviation Civile (ENAC) in 1994. He is currently completing a Ph.D. in computer science.