Collision avoidance using neural networks learned by genetic algorithms

Submitted at IEA-AEI 96

Nicolas Durand
durand@cena.dgac.fr
CENA*

Jean-Marc Alliot
alliot@dgac.fr
ENAC†

Joseph Noailles
noailles@enseeiht.fr
ENSEEIHT‡

Abstract

As Air Traffic keeps increasing, many research programs focus on collision avoidance techniques. In this paper, a neural network learned by genetic algorithm is introduced to solve conflicts between two aircraft. The learned NN is then tested on different conflicts and compared to the optimal solution. Results are very promising.

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

1 Air Traffic Control and Collision Avoidance

As Air Traffic keeps increasing, overloading of the ATC[1] system becomes a serious concern. For the last twenty years, different approaches have been tried, and different solutions have been proposed. To be short, all theses solutions fall in the range delimited by the two following extreme positions:

On the one hand, it could be possible to imagine an ATC system where every trajectory would be planned and where each aircraft would follow its trajectory with a perfect accuracy. With such a system, no reactive system would be needed, as no conflict[2] between aircraft would ever occur. This solution is close to the ARC-2000 hypothesis, which has been investigated by the Eurocontrol Experimental Center [5].

On the other hand, it could also be possible to imagine an ATC system where no trajectories are planned. Each aircraft would flight its own way, and all collisions would have to be avoided by reactive systems. Each aircraft would be in charge of its own security. This could be called a completely free flight system. The free flight hypothesis is currently seriously considered for all aircraft flying "high enough" in a quite near future.

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 each and every trajectory would be perfectly followed; there are too many parameters that can not be perfectly controlled: meteorological conditions (storms, winds, etc.), but also breakdowns in aircraft (motor, flaps, etc) 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 look like Brownian movements.

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

1. Airspace design (airways, control sectors, ...), When joining two airports, an aircraft must follow routes and beacons; these beacons are necessary for pilots to know their position during navigation and help controllers to visualize the traffic. As there are many aircraft simultaneously present in the sky, a single controller is not able to manage all of them. So, airspace is partitioned into

*Centre d'Etudes de la Navigation Aérienne
†Ecole Nationale de L'Aviation Civile
‡Ecole Nationale Supérieure d'Electronique, d'Electrotechnique, d'Informatique et d'Hydraulique de Toulouse

[1] Air Traffic Control

[2] 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

different sectors, each of them being assigned to a controller. This task 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 on the beacons inducing overloaded sectors. Traffic assignment aims at changing aircraft routes to reduce sector congestion, conflicts and coordinations.

3. Coordination planning (a few minutes ahead), This task guarantees that new aircraft entering sectors do not overload the sector.

4. Classical control in ATC centers (up to 20 mn ahead), At this level, controllers solve conflicts between aircraft.

5. Collision avoidance systems (a few minutes ahead). This level is activated only when the previous one has failed. This level is not supposed to be activated in current 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 (filter level 5) with reactive techniques. This problem solver is based on a neural network, which was built by a genetic algorithm.

2 Existing reactive techniques

The most well known concept on reactive collision avoidance is certainly the ACAS/TCAS concept. It is already implemented in its two first versions (TCAS-I and TCAS-II). 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 a the application of a sequence of filtering rules, which give the pilot a resolution advice.

A very simple technique to do reactive control has been investigated by [4]. The idea is to consider each aircraft as positive electric charges, while the destination of the aircraft is a negative charge. This way, 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 [9]. This system is slightly more complex, but 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 speed, and not only their heading. 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, as aircraft engines are easily damaged by this kind of operations.

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. Building neural networks with GA has already been done. An application quite similar was the problem of car parking described in [8]. However, our problem is definitely more complex.

3 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, when there is a conflict the heading of this aircraft (respecting operational constraint of 45 degrees maximum per 15 seconds). 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 embarked radar to detect other aircraft. Consequently, all the inputs of the neural network must be given by the radar information.

4 Using a neural network

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 will first check if both

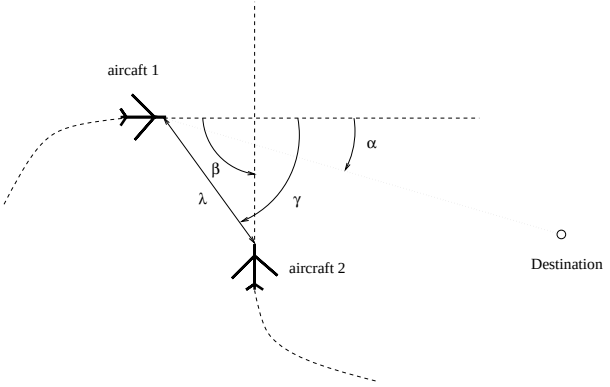

Figure 1: The neural network inputs of aircraft 1.

aircraft can connect their destination without changing their heading and without generating conflicts. In that case, we do not modify aircraft headings. If we detect a conflict in less than twenty minutes, we compute a new heading for both aircraft with the NN.

4.1 The inputs

7 inputs are used by the neural network (see figure 1) :

- The heading of the destination α and its absolute value $|\alpha|$ (in degrees).
- The distance to the other aircraft λ and its gradient $\frac{d\lambda}{dt}$.
- The bearing of the other aircraft γ (in degrees)
- The converging angle of the trajectories β.
- A bias set to 1.

4.2 The neural network structure

The neural network structure used is as simple as possible. A 3 layer 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 takes the 6 inputs described above plus the bias. The second layer holds 13 units, while the third layer holds the output unit.

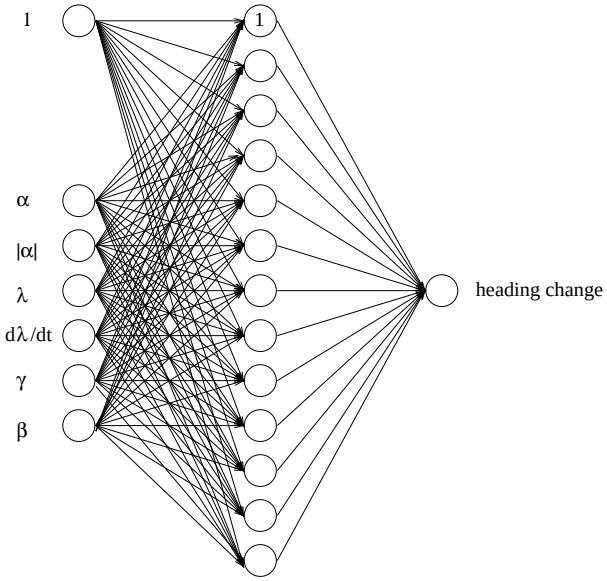

Figure 2: The neural network structure.

4.3 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 will compare the results of our network with optimal trajectories computed by LANCELOT[3] [2] to validate our hypothesis.

5 Genetic Algorithms

Figure 3 describes the main steps[4] of GAs that were used in this paper: 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 we will call *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

[3]Large And Nonlinearly Constrained Extended Lagrangian Optimization Techniques

[4]We are using classical Genetic Algorithms and Evolutionary Computation principles such as described in the literature [3, 7].

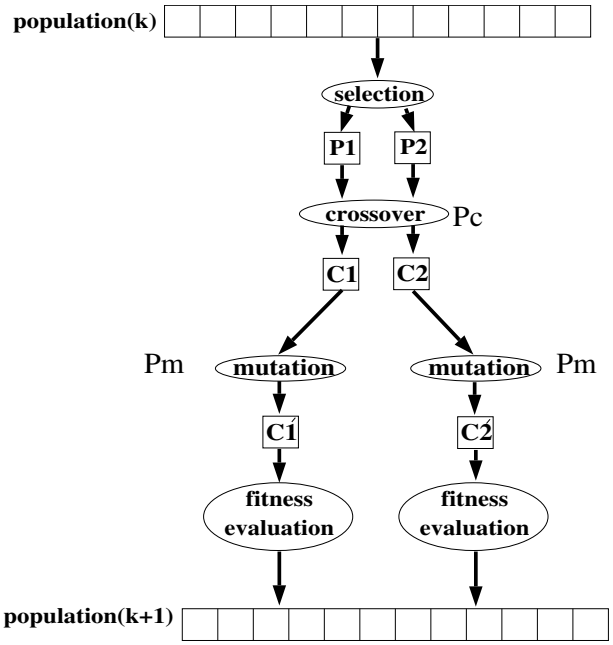

Figure 3: GA principle

at random again, and a *mutation operator* is applied, to slightly modify their structure. At this step a new population is been created and we apply the process again in an iterative way. The different steps are detailed in the following.

5.1 Coding the problem

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

5.2 Selection

A method called "Stochastic Remainder Without Replacement Selection" [3] was 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 fitness 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.

5.3 Crossover

The crossover operator we used was the barycentric crossover : 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

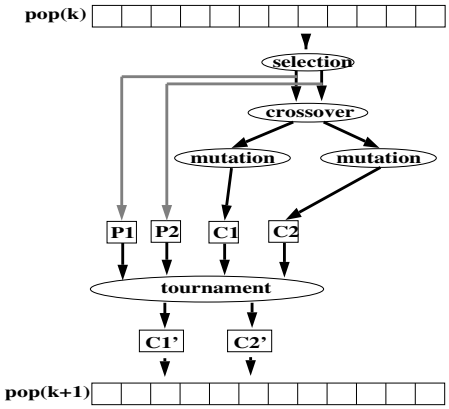

Figure 4: GA and SA mixed up

weights of $(parent_1, \alpha)$ (resp $(parent_1, 1-\alpha)$) and $(parent_2, 1-\alpha)$ (resp $(parent_2, \alpha)$). In the further applications, the crossover probability used is 60%.

5.4 Mutation

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

5.5 Simulated Annealing Tournament

GA can be improved by including a Simulated Annealing process after applying the operators [6]. 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 4. 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 [1].

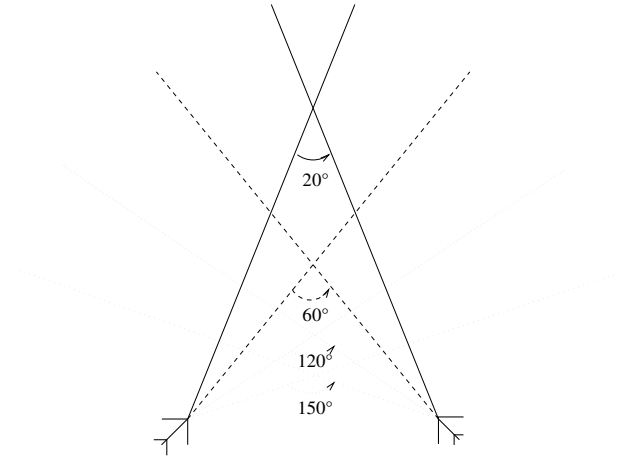

Figure 5: 4 configurations at the same speed.

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.

5.6 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.

To compute the fitness, a panel of different conflict configurations is created. The fitness is computed as follow :

$$F = \frac{1}{D} e^{-V}$$

D is the average delay due to deviations and V is the average number of conflict violations.

5.7 The learning examples

To learn the weights of the neural networks, 12 configurations were created. In each configuration, at $t = 0$ aircraft are 20 nautical miles distant.

- in 4 configurations, aircraft have the same speed and converge with different angles (20, 60, 120, 150 degrees, see figure 5).
- in 4 configurations, aircraft have different speed, their headings are calculated to generate a conflict (one aircraft speed is 500 knots and the other one is 300, 350, 400, and 450 see figure 6).

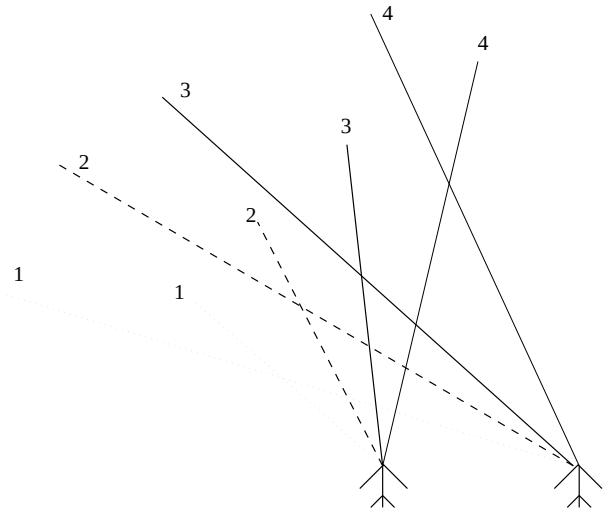

Figure 6: 4 configurations at the different speeds.

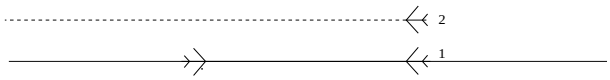

Figure 7: 2 configurations of facing aircraft.

- in 2 configurations, aircraft have opposite headings and the same speed (see figure 7).
- in 2 configurations, aircraft have the same heading but different speeds (see figure 8).

Because of symmetries, these 12 configurations summarize all the situations that can happen. We will call "positive configuration" (see figure 9) a configuration in which the angle between the slowest aircraft and the fastest is positive. When a "negative configuration" occurs, the symmetrical positive configuration is used in the neural network to calculate the deviation. Therefore, some of the inputs and the output are given the opposite sign.

6 Numerical results

The neural network was learned using the following parameters :

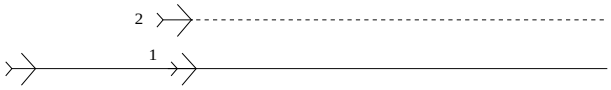

Figure 8: 2 configurations of facing aircraft.

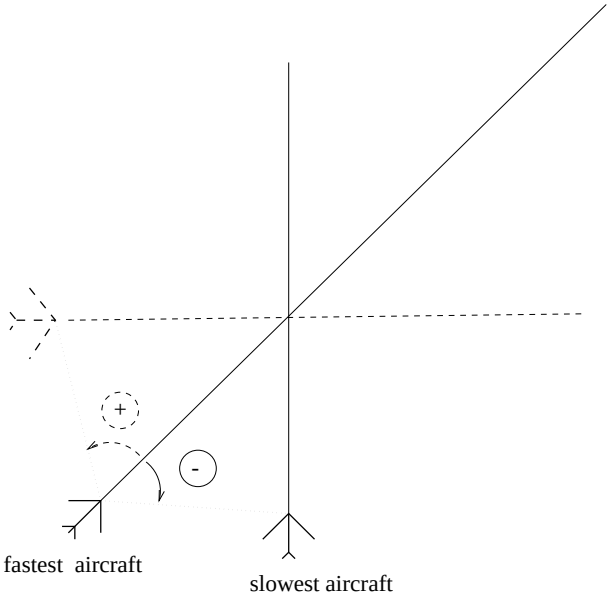

Figure 9: Symmetrical configurations.

number of generations : 500
population elements : 500
percentage of crossover : 60
percentage of mutation : 15
simulated annealing for crossover : yes

Optimal solutions to the different configurations be 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). It is then not usable to control aircraft in real time. However, it is interesting to compare optimal solutions found by LANCELOT to solutions learned by the neural network. Learned solutions are obviously less optimal, but the loss of optimality is not significant (the delay induced by the neural network is never more than twice the minimal delay, which is generally very small).

The configurations used to compare the neural network to optimal solutions are not learned configurations. We want also to validate the capacity of the NN to generalize to non-learned situations :

- Figure 10 gives an example of conflict at 90 degrees in which aircraft have the same speed. Neural network and optimal solution are similar.

- Figure 11 gives an example of conflict at 15 degrees in which 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 gives a solution that is good and robust.

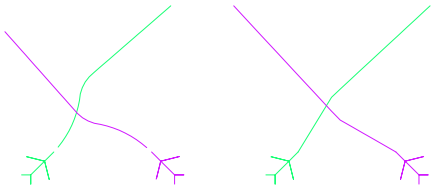

Figure 10: Neural network solution (left), optimal solution (right).

- Figure 12 gives an example of aircraft at different speeds (400 and 500 knots) with crossing at a small angle (30 degrees). The neural network solution is very similar to the optimal solution.

- Figure 13 gives an example of aircraft crossing on the same route. This problem is easy to solve and solutions are similar.

- Figure 14 gives an example of aircraft flying on parallel routes at different speeds. This problem is easy to solve and solutions are similar.

7 Conclusion

Using a simple neural network to solve a conflict between 2 aircraft have given very good results. It was shown above that the neural network could be easily learned by a genetic algorithm without knowing the optimal solutions. The next step of this work will consist in extending the problem to conflicts involving more than 2 aircraft. As the problem becomes very combinatory, some hypothesis will probably have to be made to limit the size of the neural network. The third step will be to integrate climbing an descending aircraft in the model and to generate vertical manœuvres. The results presented above should be very soon used in a Test Bench to check their validity on real traffic.

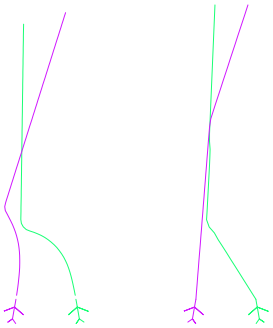

Figure 11: Neural network solution (left), optimal solution (right).

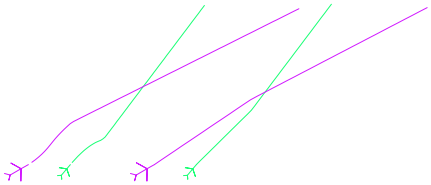

Figure 12: Neural network solution (left), optimal solution (right).

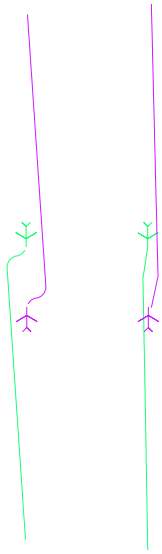

Figure 13: Neural network solution (left), optimal solution (right).

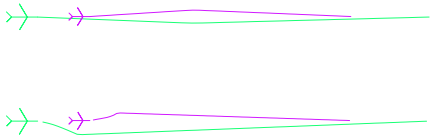

Figure 14: Neural network solution (down), optimal solution (up).

List of Figures

References

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

[2] 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.

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

[4] 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.

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

[6] 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.

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

[8] Marc Schoenauer, Edmund Ronald, and Sylvain Damour. Evolving nets for control. Technical report, Ecole Polytechnique, 1993.

[9] Karim Zeghal. *Vers une théorie de la coordination d'actions, application à la navigation aérienne.* PhD thesis, Universite Paris VI, 1994.