Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)

Ant Colony Optimization for Air Traffic Conflict Resolution

Nicolas Durand
DTI R&D/POM
DSNA
Toulouse, France
durand@aviation-civile.gouv.fr

Jean-Marc Alliot
DTI R&D
DSNA
Toulouse,France
alliot@aviation-civile.gouv.fr

***Abstract*— The n aircraft conflict resolution problem is highly combinatorial and can be optimally solved using classical mathematical optimisation techniques only for small problems involving less than 5 aircraft. This article applies an Ant Colony Optimization (ACO) algorithm in order to solve large problems involving up to 30 aircraft. In order to limit the number of pheromone trails to update, a n aircraft conflict resolution problem is not modeled by a single ant but by a bunch of n ants choosing their trajectories independantly. A relaxation process is also used in order to be able to handle difficult conflicts for which partial solutions can help finding a path toward the optimal solution. Two different sizes of a toy problem are solved and presented.**

Keywords:

Ant Colony Optimization, Metaheuristic, Air Traffic Conflict Resolution

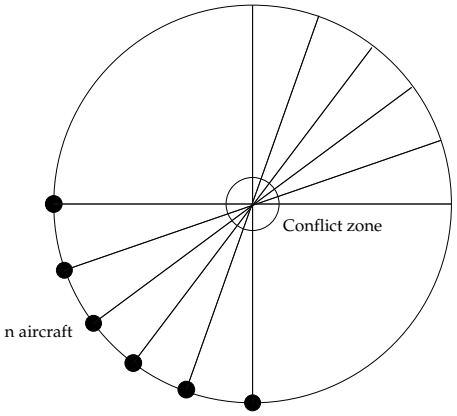


Fig. 1. n aircraft conflict problem

I. Introduction

Aicrcaft conflict resolution is highly combinatorial. It cannot be solved using classical optimization techniques and realistic models when the number of aircraft involved exceeds 4 or 5. The most efficient classical algorithm was developped by Palatino, Feron and Bicchi [PFB02] using mixed integer programming. They were able to handle up to 15 aircraft on a toy problem, but the hypotheses on trajectories were totally unrealistic because they assumed constant speeds and climbing rates as well as no uncertainty. With stochastic optimization methods such as Evolutionnary Algorithms, Durand and Alliot [DA97] were able to handle conflicts involving up to 30 aircraft by taking advantage of the partial separability of the problem [DA98].

Algorithms such as Branch and Bound methods using Interval analysis [MÓ98], or semi-definite programming [FMF01], [Dod99] succeeded in solving smaller toy problems, but each time the hypothesis on the trajectories (constant speeds, constant climbing rates, no uncertainty) remained unsuitable in a real context. Other approaches using potential fields models [TPS98], [GT00], neural networks [DAN96], [DAM00], and linear programming [MÓ94] did not give better results and also relied on unrealistic assumptions on trajectories predictions.

In this article, a new approach using Ant Colony Optimization (ACO) is introduced and adapted in order to solve large conflicts. ACO was introduced by Marco Dorigo [DMC96], [DC99] in the 90's. ACO takes inspiration from the behaviour of some ant species. These ants deposit pheromones on the ground in order to mark some favorable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving problems.

In part two of the article, the problem modeling is discussed. In part three the algorithm is detailed. An improvement of the algorithm in order to deal with difficult problems is detailed in part four. In part five the algorithm is tested on a 5 and 30 aircraft conflict toy problem. Further work to be done is discussed in the conclusion.

II. Problem modeling

The toy problem used in this article is described on figure 1. n aircraft are located on a circle of radius R and flying to the center of the circle with the same speed. Their destination is the point of the circle located at the other end of the circle diameter they are flying on. This problem is known to be difficult to solve ([Dur96]) because each aircraft is in conflict with every other aircraft. The objective is to find new trajectories for every aircraft that solve every conflict and minimise the extra distance flown.

As long as humans pilot aircraft, the maneuvers orders given to the aircraft to solve conflicts must remain simple

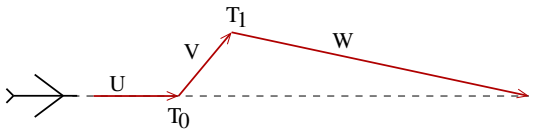


Fig. 2. Maneuver modeling

to understand and execute. This means that in the modeling, a single maneuver is given per aircraft. Time is discretized. The maneuver starts at some time T_1 and ends at some time T_2. In this article, conflicts are solved horizontally : a heading change of 10, 20 or 30 degrees right or left is given to the aircraft (see figure 2).

An aircraft trajectory can be modeled by a graph. Time is discretized in n_t timesteps. The graph of the aircraft positions can be defined as follows: each node represents a time and an aircraft position. The transition from position i to position $i+1$ is represented by an edge on which ants deposit pheromones.

An aircraft can be successively in three states. An aircraft is in state U before any maneuver. When a maneuver is started at T_1, it is in state V. Finally, when the maneuver is ended at T_2, aircraft is in state W. If U_i, V_i and W_i are the number of possible aircraft positions respectively in state U, V and W at time i, then:

$$\begin{aligned} U_{i+1} &= U_i \\ V_{i+1} &= V_i + 6 \\ W_{i+1} &= V_i \end{aligned}$$

with $U_1 = 1$, $V_1 = 6$ and $W_1 = 0$. It can be easily deduced that $U_i + V_i + W_i = 12\,i - 5$.

If we consider that one ant represents one conflict solution of an ACO (see figure 3), then for n_a aircraft and n_t timesteps, the number of possible trails at time i is $(12\,i-5)^{n_a}$, and the total number of possible trails is $(12\,n_t - 5)^{n_a}$. For $n_a = 5$ and $n_t = 10$, more than 10^{10} trails can be obtained.

The modeling presented in this article considers a bunch of n_a ants to solve a n_a conflicting aircraft problem. Ants are treated independantly, except to calculate the quantity of pheromones to deposit which depends on the number of conflicts each ant of the bunch has been able to solve. This modeling reduces the number of trails to update to $n_a\,(12\,n_t - 5)$. For $n_a = 5$ and $n_t = 10$, the number of trails to update is 575 which is far less excessive than 10^{10}. For $n_a = 30$ and $n_t = 20$ the number of trails to update is only 7050 instead of 10^{71} with the previous modeling.

III. Algorithm description

The ACO algorithm, is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The original idea was introduced by [DMC96], [DC99]. First algorithms were tested on the Travelling Salesman Problem. The algorithm mimics the behavior of ants seeking a path between their colony and a source of

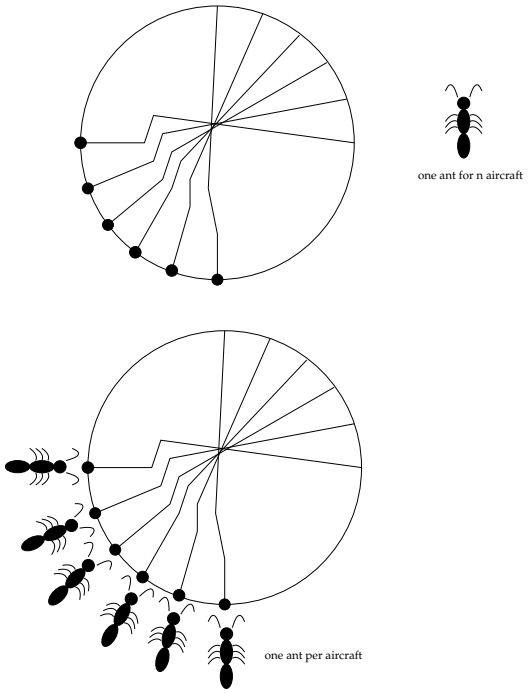


Fig. 3. one ant per cluster or one ant per aircraft

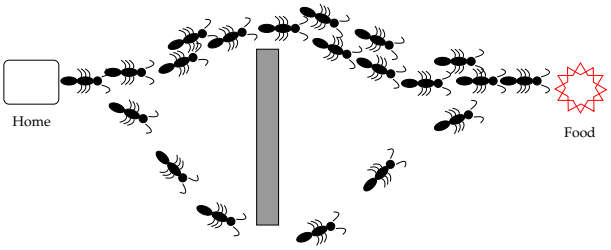


Fig. 4. Best path choice to get some food

food (see figure 4). The idea has since been diversified to solve a wider class of numerical problems.

Ants use the environment as a medium of communication. They exchange information by depositing pheromones. The information exchanged has a local scope. On the Travelling Salesman Problem, the first ants choose their path randomly and deposit all the more pheromones since the path chosen is short. New ants then choose their path taking into account the amount of pheromones they find locally. The more pheromones they find on a given path, the likelier they are to take it. A dissipation of the path by evaporation of the pheromones prevents the algorithm from premature convergence.

The ACO used in this article is a classical ACO as presented by Dorigo. The only difference is that an ant is replaced by a bunch of n_a ants representing the n_a aircraft. Each ant of a bunch represents an aircraft. An ant can be in three different states as shown on figure 5.

- Before any maneuver the ant ins in state U
- After T_0, it changes its heading and moves to state V

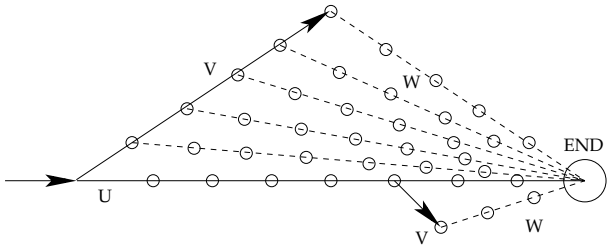


Fig. 5. Graph modeling

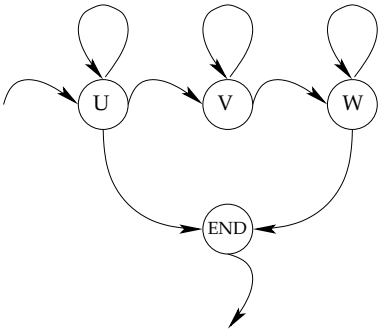


Fig. 6. Possible transitions

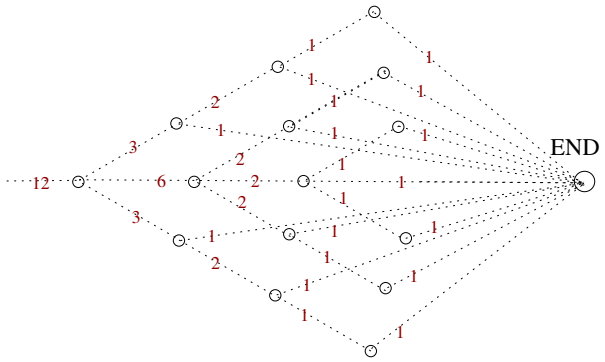


Fig. 7. Initial amount of pheromones on the graph

- After T_1, it changes its heading and moves to state W

At each node of the graph representing the possible trajectories of an aircraft, the ant chooses the next node with a probability depending on the quantity of pheromones left on the edge connecting the two nodes. The trajectories are then tested in order to check the existing conflicts. Ants representing conflicting aircraft do not deposit any pheromones whereas ants representing conflict free aircraft deposit pheromones. The quantity of pheromones deposited decreases with the delay due to the aircraft maneuver.

The graph of possible paths is built in order to accept a maximum delay for each aircraft. At the beginning of the algorithm, initial pheromones are spread on the graph in order to ensure an equal probability for each path to be chosen. Figure 7 gives an example of the distribution of initial pheromones that ensures equal chance to every path. In this simple example, aircraft can turn left or right (30 degrees) or go straight and only a few steps are represented. The amount of pheromones on each edge is thus proportional to the number of possible path remaining after passing through this edge. Starting from the END, a state W is given a unit of pheromones, and then pheromones are added at each node in order to fill the whole graph.

If n_a is the number of aircraft, a bunch of $p \times n_a$ ants is created in order to represent each aircraft at each generation. Thus, the total number of ants is $p \times n_a^2$ (in the example, $p = 10$).

Each path is given a score. The smaller the score, the better the path is. Because the straight line is the shortest path, getting through state U does not change the score. State V adds 2 points and state W adds 1 point. This scoring gives an advantage to maneuvers starting late: when aircraft are in state W the score increases whereas in state U it does not.

At each node an ant chooses the next edge with a probability depending on the quantity of pheromones left on the next edge.

If the ant is in conflict with another ant, it does not deposit any pheromones. However, when there is no conflict, it leaves behind it a quantity of pheromones equal to

$$\Delta\tau = \frac{n_a - n_{\text{out}}}{n_a} \cdot \frac{\tau_0}{s_{\text{path}}}$$

where n_{out} is the number of "lost" ants, τ_0 the original quantity of pheromones, and s_{path} the score of the path followed by the ant. This amount takes into account the number of ants that finally found a valid path.

After each generation, and before starting a new generation an evaporation principle is applied on the existing trails. The amount of pheromones is decreased by $x\%$ (in the examples $x = 10\%$) at the end of each generation.

The algorithm ends when the score obtained by each bunch of ants representing each aircraft does not decrease for a while or when the time allowed for the algorithm runs out.

IV. Algorithm Improvement: Constraint Relaxation

In high density areas, conflicts might become difficult to solve and it may happen that a random generation of maneuvers cannot solve any conflict. This means that none of the ants might be able to solve every conflict. In such a case, there is no way to find even a bad solution for the problem. We propose to relax the conflict resolution constraint during the first generations in order to help the algorithm to find solutions with a small number of conflicts remaining. When solutions are found for a certain number of ants, the constraint is reinforced in order to move toward solutions that solve more conflicts and so on. For example, at the first generation of the algorithm, we count the number n_c of ants having less than c conflicts for $c = 0, 1, 2, 3...(n_a - 1)$. Let us define r as the maximum value of $\frac{n_a}{n_c}$ rounded to the higher integer. r gives the number of allowed conflicts per ant at the first generation. Each time the number of ants having less than r conflicts is higher than $\frac{n_a}{r}$ then r is reduced by one unit. This is repeated as long as $r > 0$.

There are lots of ways to choose r and to make it decrease. The choice made in this article is empirical and further work needs to be done to check it on different conflict sizes and configurations. Different strategies need to be compared.

V. Results

In this section, a difficult toy problem is solved with the ACO algorithm described in the previous section. Figure 8 shows the best solution obtained after 18, 46 and 105 iterations of the algorithm for the 5 aircraft problem. On this figure, we can see how the algorithm is able to deal with the combinatorial characteristics of the problem because the different solutions found at different steps of the algorithm do not give the same combinations of maneuvers to the aircraft. The scores obtained decrease with the generations. At generation 18 the score is 89, at generation 46 it is 78, and at generation 105 it is 50.

Figure 9 shows the best solution obtained for a 30 aircraft problem at different steps of the algorithm. The initial picture shows the best solution at the first generation. Only 9 ants having 4 conflict or less are represented. The conflict constraint is reinforced because $\frac{30}{9} \leq 4$. Only ants having less than 3 conflicts now survive. At generation 14, 13 ants are having less than 3 conflicts. The conflict constraint is reinforced again because $\frac{30}{13} \leq 3$. At generation 15, despite the reinforcement 13 ants are still having less than 2 conflicts. This number increases to 20 at generation 44 and the conflict constraint is reinforced again because $\frac{30}{20} \leq 2$. At the next generation, the number of ants having less than 1 conflict is still 20 and increases to 30 at generation 47. The no conflict constraint is applied at this step because $\frac{30}{1} \leq 1$. The first solution of the problem is found at generation 48. The solution is then improved and the ending criteria occurs at generation 65.

VI. Conclusion and further work

The n aircraft conflict resolution problem is very complex and and cannot be solved with classical optimization tools when n is large. Furthermore, classical optimization algorithms generally require trajectory modelings that do not match with real constraints. This article presents a new approach for solving the problem using Ant Colony Optimization. Trajectories are modeled by a simple graph and the algorithm is adapted in order to deal with a reasonable number of trails: each aircraft is represented by a bunch of ants that optimizes its trajectory. A relaxation principle is also added in order to help the algorithm find solutions when the conflict is complex to solve. When the algorithm starts to converge, it becomes possible to reinforce the constraint to find a no conflict solution.

The algorithm has not been tested on real examples yet and a lot of values have been empirically chosen. However the algorithm is able to deal with very large problems as it was shown in an example involving 30 aircraft. The algorithm must now be compared to other efficient methods such as evolutionnary algorithms and tested on other examples with

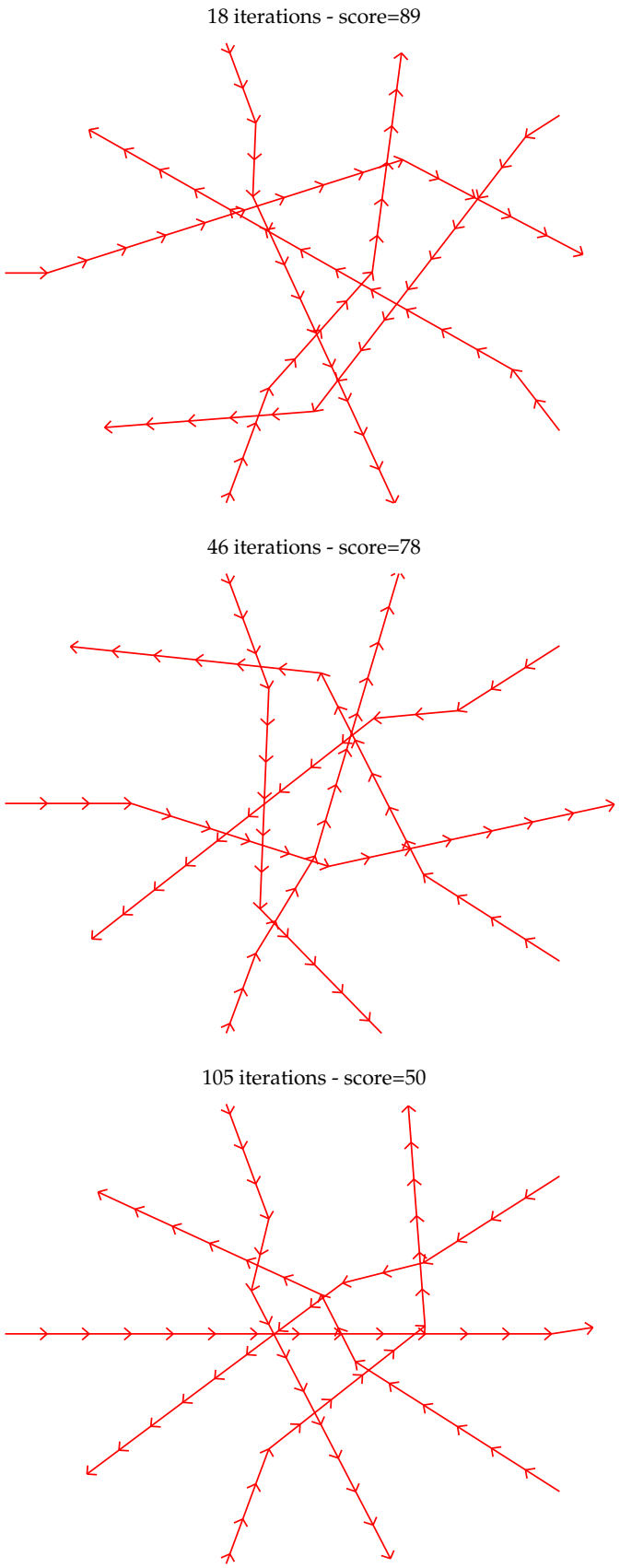
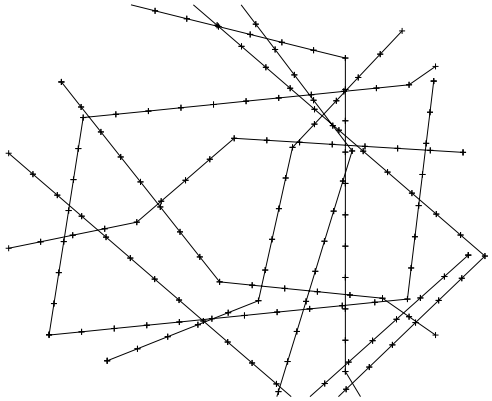
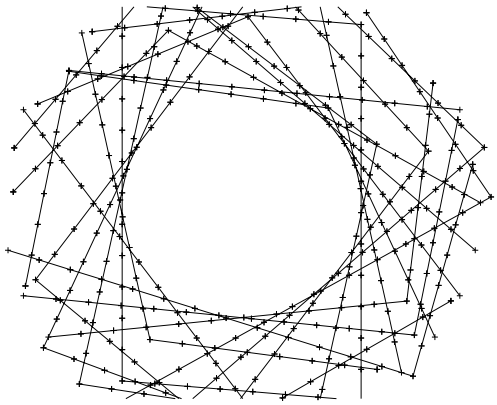


Fig. 8. Example of 5 aircraft conflict resolution

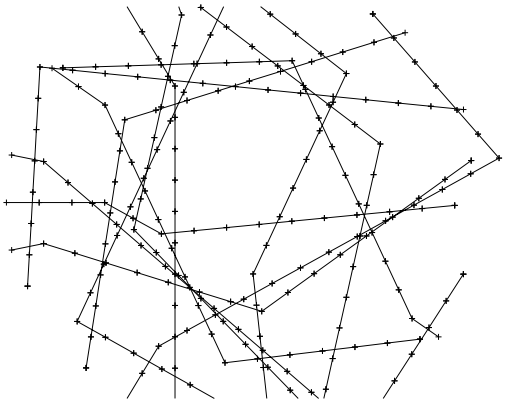
generation: 0 - 4 conflicts max - 9 aircraft



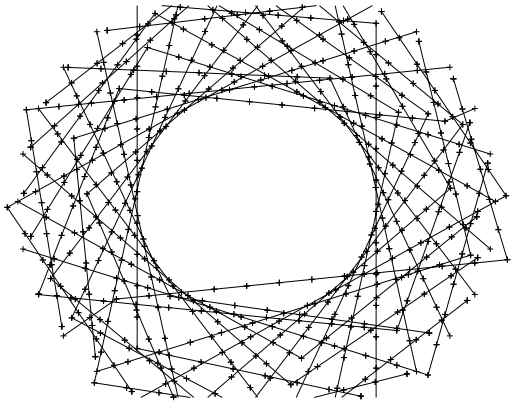
generation: 45 - 1 conflicts max - 20 aircraft



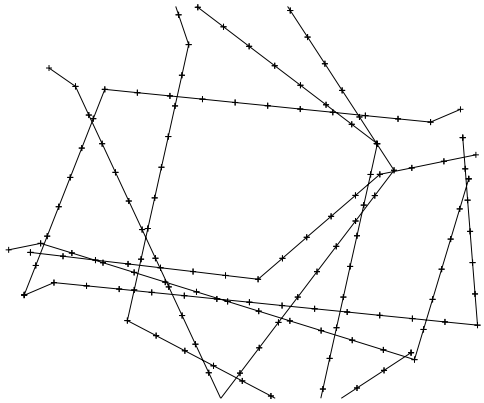
generation: 14 - 3 conflicts max - 13 aircraft



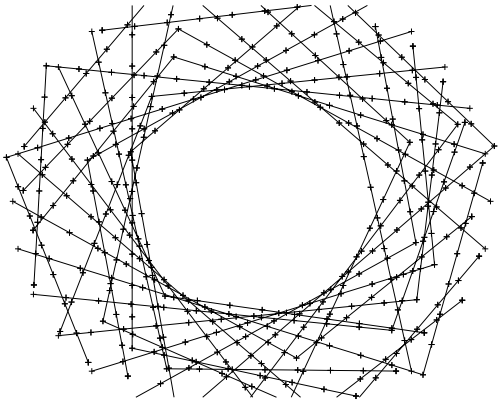
generation: 47 - 1 conflicts max - 30 aircraft



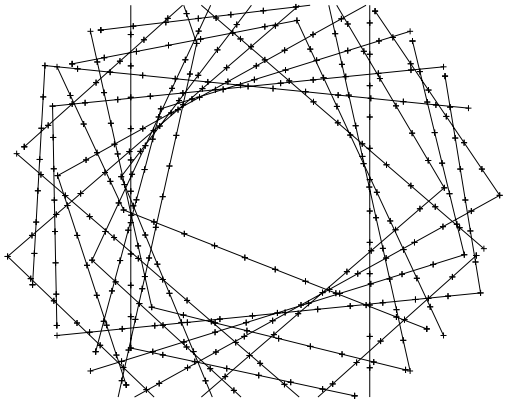
generation: 15 - 2 conflicts max - 13 aircraft



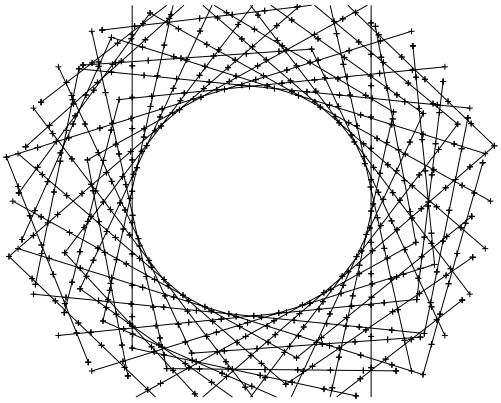
generation: 48 - 0 conflicts max - 30 aircraft



generation: 44 - 2 conflicts max - 20 aircraft



generation: 65 - 0 conflicts max - 30 aircraft



different numbers of aircraft and conflicts. It could then be tested on real data with a fast time simulator.

REFERENCES

[DA97] N. Durand and J.M. Alliot. Optimal resolution of en route conflicts. In *Procceedings of the 1rst USA/Europe Seminar*, 1997.

[DA98] Nicolas Durand and Jean-Marc Alliot. Genetic crossover operator for partially separable functions. In *Genetic Programming*, 1998.

[DAM00] N. Durand, J.M. Alliot, and F. Medioni. Neural nets trained by genetic algorithms for collision avoidance. *Applied Artificial Intelligence*, 2000.

[DAN96] N. Durand, J.M. Alliot, and J. Noailles. Collision avoidance using neural networks learned by genetic algorithms. In *Ninth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Fukuoka*, 1996.

[DC99] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Congress on Evolutionary Computation*, 1999.

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41, 1996.

[Dod99] Pierre Dodin. Résolution de conflits via la programmation semie-définie. Master's thesis, Paris VI, 1999.

[Dur96] Nicolas Durand. *Optimisation de trajectoires pour la résolution de conflits en route*. PhD thesis, INPT, 1996.

[FMF01] E. Frazzoli, Z.H. Mao, and E. Feron. Aircraft conflict resolution via semidefinite programming. *AIAA Journal of Guidance, Control and Dynamics*, 2001.

[GT00] R. Gosh and C. Tomlin. Maneuver design for multiple aircraft conflict resolution. In *American Control Conference*, 2000.

[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.

[MÓ98] Frédéric Médioni. *Méthodes d'optimisation pour l'évitement aérien : systèmes centralisés, systémes embarqués*. PhD thesis, Ecole Polytechnique, 1998.

[PFB02] L. Pallottino, E. Feron, and A. Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):3–11, 2002.

[TPS98] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: a case study in multi-agent hybrid systems. In *IEEE Transactions on Automatic Control*, 1998.

BIOGRAPHY

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) from 1992 to 2007 and holds a Ph.D. in computer Science (1996). He is currently in charge of the air traffic Planning, Optimization and Modeling team of the R&D department of DSNA.

Jean-Marc Alliot graduated from the Ecole Polytechnique de Paris in 1986 and from the Ecole Nationale de l'Aviation Civile (ENAC) in 1988. He also holds a Ph.D. in Computer Science (1992). He has been in charge of the global optimization laboratory (LOG) of CENA and ENAC in Toulouse from 1996 to 2007. He is currently deputy director of the R&D department of DSNA.