Benchmarking conflict resolution algorithms

Charlie Vanaret, David Gianazza, Nicolas Durand, Jean-Baptiste Gotteland

Benchmarking Conflict Resolution Algorithms

Charlie Vanaret, David Gianazza, Nicolas Durand and Jean-Baptiste Gotteland
ENAC/MAIAA, IRIT/APO, Toulouse, France
{ *vanaret*, *gianazza*, *durand*, *gottelan* }@recherche.enac.fr

***Abstract*—Applying a benchmarking approach to conflict resolution problems is a hard task, as the analytical form of the constraints is not simple. This is especially the case when using realistic dynamics and models, considering accelerating aircraft that may follow flight paths that are not direct. Currently, there is a lack of common problems and data that would allow researchers to compare the performances of several conflict resolution algorithms.**

The present paper introduces a benchmarking approach that can provide researchers with common problems, in order to compare the performances of several conflict resolution algorithms. A comparison between three resolution methods is drawn over several problems and highlights assets and weaknesses for each of them.

I. Introduction

Solving aircraft conflicts is a highly combinatorial problem that has never been solved with classical methods under realistic hypotheses yet. Two types of approaches exist: *Autonomous* (or distributed) approaches in which each aircraft is equipped with on-board control and has a limited visibility of the surrounding aircraft, and *centralized* approaches in which the air traffic controllers have a global visibility of the whole traffic.

Autonomous approaches were designed in the 1990s as parts of the Free-Flight concept, allowing relaxation of routing restrictions and aiming at selecting aircraft trajectories in real time ([1], [2], [3]). The approach introduced in [1] is based on force fields for handling multiple mobiles. The coherence of movements is provided by a pairwise definition of a sliding force. No guarantee of consistency is given for more than 2 aircraft. FACES [2] is a coordinated on-board conflict solver for Free-Flight airspace, in which coordination is ensured by a token allocation strategy. Conflicts are solved by an A^* algorithm after aircraft have been sorted by priority level. This approach has been widely tested in [4] for few aircraft and ensures local optimality, but has no guarantee of success. An algorithm for detecting and analyzing potential en route conflicts is designed in [3] to produce a set of probable future conflicts, and assist the en route sector controller in efficiently solving these conflicts. Conflicts are detected via comparisons of trajectories at closely spaced time instants.

The only centralized approaches so far that can efficiently solve conflicts involving more than 20 aircraft and find a global optimum, are mixed integer programming ([5], [6]) and evolutionary algorithms ([7], [8], [9]). In [5] the hypotheses about the aircraft speed are strong and hardly realistic within an operational research framework: constant velocities for lateral avoidance resolution, or instantaneous changes of velocity for speed resolution. In [7] a genetic algorithm uses an evolving population of candidate solutions that encodes sideways maneuvers for aircraft that are initially in conflict with each other. Their modeling relies on discretized variables: start and end times of maneuver are chosen with a fixed time-step, and deviation angles are chosen among discrete values. A similar modeling is used in [9] with an ant colony optimization algorithm. Evolutionary algorithms show very convincing results, under relatively realistic hypotheses allowing to take into account uncertainty about the aircraft's velocities, contrary to mixed integer programming in [5] (constant velocities, synchronised maneuvers). However, the facilities nowadays available in control centers and aircraft are not designed to handle temporally-defined maneuvers.

In this paper, we introduce a set of benchmark problems for conflict solving, and then compare three different optimization algorithms on this common benchmark. Conflict solving is seen as a global optimization problem under constraints. The objective is to follow as much as possible the initial trajectories while not violating the separation constraints. The proposed benchmarking approach is a "black-box" one, in that evaluating the separation constraints requires to simulate the trajectories. However, the description of the benchmark problems used in this paper can be made available to other researchers in the form of simple XML files[1].

As in [5], a simple model with lateral maneuvers only was used. It could easily be extended to vertical and

[1]MAIAA website is under construction

speed maneuvers. We applied optimization methods that, unlike [7], operate on continuous variables: The start, deviation angle and length of lateral maneuvers can take any floating-point value within given bounds. We developed a small simulator implementing these features, that takes the XML description of the benchmark problems as input. As an illustration of the proposed benchmarking approach, three different optimization methods were compared on our benchmark: a genetic algorithm with a floating-point encoding, a particle swarm optimization method, and a differential evolution algorithm. The results obtained provide novel insights on why some of these methods perform better than others when applied to conflict resolution problems.

The rest of the paper is organized as follows. The standard formalization of optimization problems is given in Sec. II-A. In Sec. II-B, our conflict resolution problem is formalized as a constrained optimization problem. The benchmark approach and the problems considered are then introduced in Sec. III. The optimization algorithms compared in this study are detailed in Sec. IV. In Sec. V, results are examined and interpretations are proposed to explain the algorithms' performances.

II. Conflict solving as an optimization problem

A. Usual optimization problems

Standard optimization problems are usually formalized as follows (considering here a *minimization* problem):

$$\begin{aligned} \min_{x \in \boldsymbol{D}} \quad & f(x) \quad \text{subject to} \\ & g_k(x) \leq 0, \quad k = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{1}$$

where $f(x) : (\boldsymbol{D} \subset \mathbb{R}^n) \rightarrow \mathbb{R}$ is the objective function to be minimized over the variable x, $g_k(x) \leq 0$ are inequality constraints, and $h_i(x) = 0$ are equality constraints.

Several benchmarks are available in the public domain, allowing the researchers to compare the performances of optimization algorithms[2]. Such benchmarks are easy to implement when the analytical form of the objective function and constraints are known.

[2]See for example the CEC Competition on Large Scale Global Optimization, where a test suite of optimization problems is available on the web site http://staff.ustc.edu.cn/~ketang/cec2012/lsgo_competition.htm

B. The conflict resolution problem

Solving conflicts among aircraft trajectories is also an optimization problem, where one aims at minimizing a global cost based on the deviations from the initial trajectories, while avoiding loss of separation.

In this paper, we shall consider *lateral* maneuvers only. A lateral deviation from the initial route is modeled as a triplet (d_i, α_i, l_i), where d_i is the distance along the route at which the maneuver of flight i begins, α_i is the deviation angle, and l_i is the length of the deviation segment. At the end of this deviation segment, the aircraft returns to its initial trajectory with a new deviation of angle (-2α). If this "resume nav" segment does not intercept the initial route, then the aircraft flies directly toward its destination. The modeling provides a sequence of geographical points (Fig. 1).

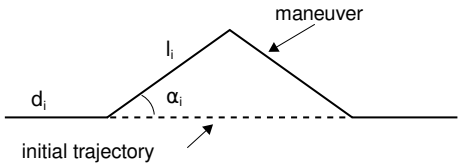


Figure 1: Turning point modeling

To reduce the deviation induced by a maneuver, both its angle and its length are to be minimized. On the contrary, the distance at which the maneuver starts is to be maximized, in order to avoid premature maneuvers that could be reduced or canceled in future resolutions, with less uncertainties on the positions of aircraft.

Thus, in this paper, the cost of a lateral deviation $x_i = (d_i, \alpha_i, l_i)$ is defined as follows, denoting $\overline{v}$ the upper bound of a variable v:

$$cost(x_i) = \left(\frac{\overline{d_i} - d_i}{\overline{d_i}}\right)^2 + \left(\frac{\alpha_i}{\overline{\alpha_i}}\right)^2 + \left(\frac{l_i}{\overline{l_i}}\right)^2 \tag{2}$$

Considering n aircraft and a vector of lateral deviations $x = (x_1, \ldots, x_n)$, we have chosen to minimize the following objective function:

$$f(x) = \sum_{i=1}^{n} cost(x_i) \tag{3}$$

Finally, the conflict resolution problem can be formalized as follows:

$$\begin{aligned} \min_{x \in \boldsymbol{D}} \quad & f(x) \quad \text{subject to} \\ & dist(p(x_i, t), p(x_j, t)) > S_h \quad \forall i \neq j, \forall t \end{aligned} \tag{4}$$

where $dist$ is the Euclidean distance, $p(x_i, t)$ is the position of aircraft i at time t, given $x_i = (d_i, \alpha_i, l_i)$

the lateral deviation from the initial route, and S_h is the standard horizontal separation (Fig. 2). The domain $\boldsymbol{D}$ of the variable x is defined by the lower bounds $(\underline{d_i}, \underline{\alpha_i}, \underline{l_i})$, and upper bounds $(\overline{d_i}, \overline{\alpha_i}, \overline{l_i})$ for each lateral deviation $x_i = (d_i, \alpha_i, l_i)$.

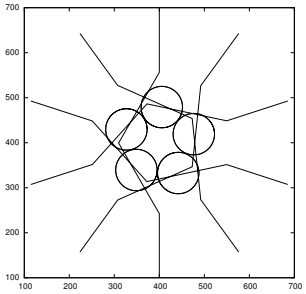


Figure 2: Horizontal separations between 5 aircraft

III. Benchmarking conflict resolution algorithms

A. A black-box approach

To address the lack of comparison material, we propose a black-box approach: a library encapsulating a light air traffic simulator provides a function $conflicts$ returning the violations of the separation constraints, given an input vector x of lateral deviations: $conflicts(x)$ returns a matrix C, where each element $C_{i,j}$ represents the duration of the conflicts occurring between aircraft i and j. A conflict is here defined as a continuous time segment during which the separation constraint defined in 4 is violated. As several conflicts may occur between two aircraft, $C_{i,j}$ is a list of time durations. The matrix C can thus be used to assess both the number and duration of conflicts among aircraft trajectories.

Conflicts can be detected by considering linear route segments and by solving an equation of the second degree when assuming constant speeds, or an equation of the fourth degree when assuming variable speeds and constant accelerations. In our case, however, the trajectories are predicted with a timestep δt seconds, and conflicts are simply detected pairwise by considering the distances between aircraft at each time step.

B. Features of the light simulator

The light simulator, encapsulated in the library written in OCaml, introduces new features when compared to other previous simulators like CATS/OPAS ([4]): aircraft operations (accelerate, follow track, etc.) are modeled as edges of a directed acyclic graph, where the nodes are triggering events (speed captured, or waypoint captured, for example). Although the simulation loop has a default time step δt that is constant, a smaller variable timestep is applied when an event is triggered between the current step at t and the next expected step at $t + \delta t$.

This feature makes it possible to simulate lateral deviations (d_i, α_i, l_i) where the values of d_i, α_i, l_i are each taken in the continuous space $\mathbb{R}$, instead of using discrete values as in previous works ([7]). Thus, continuous optimization methods can be directly applied to the conflict resolution problem.

The aircraft flight plans and initial states are read from an XML file, as well as the simulation parameters and the lower and upper bounds of the variables describing the trajectory deviations. A small program giving an example on how to build and save various problems into an XML file is also provided.

C. Experimental setup

Two problems have been considered to simulate a situation with n aircraft: *Circle_n* (Fig. 3) and *Rand_n* (Fig. 4). Dimensions are in NM.

a) Circle_n: The aircraft are arranged around a circle and their trajectories converge simultaneously toward the center. Two optimal solutions exist: in the first one all aircraft turn right, in the second one all aircraft turn left (roundabout configurations).

The fact that the problem is highly constrained can be shown by randomly generating one million situations with n aircraft, and by enumerating the situations without conflict. For $n = 5$, 497 situations are without conflict. For $n \geq 8$, no generated situation is without conflict.

In spite of being highly combinatorial, *Circle_n* remains a symmetric problem. To induce more complex behaviors in the trajectories, we introduce below another initial configuration with random flight directions.

b) Rand_n: The aircraft are arranged around a circle and each trajectory is a straight line whose direction is randomly chosen with an angle $\in [-\frac{\pi}{6}, \frac{\pi}{6}]$ to the diameter of the circle. The end point belongs to the circle as well.

IV. Conflict resolution algorithms

Metaheuristics are optimization algorithms that try to iteratively improve a candidate solution according to an adaptation criterion. Its value is determined by Eq. 3. Three population-based metaheuristics were compared in this paper: genetic algorithms, particle swarm optimization and differential evolution algorithms. Individuals

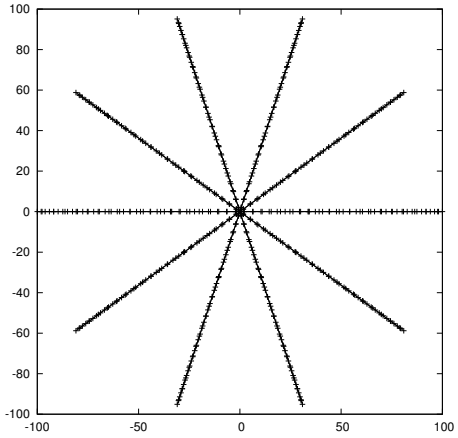


Figure 3: Flight directions of *Circle_10*

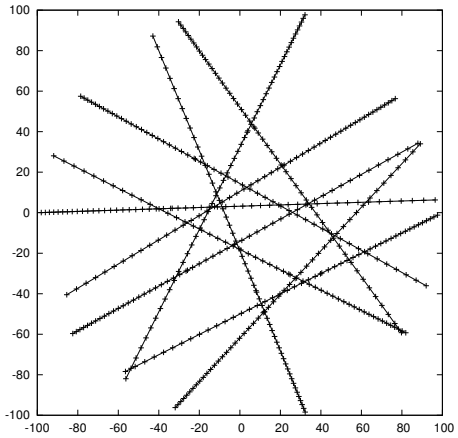


Figure 4: Flight directions of *Rand_10*

encoding solutions are moved around in the search-space by using variation and selection operations.

Each individual x in the population designates a situation with n aircraft, and is encoded by a vector $(x_1, \ldots, x_i, \ldots, x_n)^T$ representing the sideways maneuvers of the n aircraft, with $x_i = (d_i, \alpha_i, l_i)$.

A. Genetic algorithms

Genetic algorithms (GA) mimic the process of natural evolution in that they model operations such as inheritance, mutation, selection, and crossover [10]. Individuals gradually evolve at each generation and the population is partially replaced after crossover (with probability p_c) and mutation (with probability p_m) operations (Alg. 1).

In the following, p_c is set to 0.6 and p_m to 0.2. N is the population size, and is set to 500 for *Rand_20* and *Rand_30*, and to 1000 for *Circle_25*, *Rand_40* and *Rand_50*.

Algorithm 1 Genetic algorithm

```
Randomly initialize each individual
Evaluate each individual
while termination criterion is not met do
    Apply scaling and sharing to the cost values
    Replicate best-fit individuals
    Apply p_c N crossovers and p_m N mutations
    Evaluate the cost of newly generated individuals
end while
Return the best individual(s) of the population
```

Our implemented genetic algorithm benefits from two classical improvements: *scaling* and *clusterized sharing*. The selection process then operates on the image of the cost value under the scaling and sharing functions. *Scaling* aims at modifying the cost values to artificially reduce or amplify gaps between individuals' cost values, thus enhancing the exploration of the search-space. *Sharing* prevents the gathering of individuals around a prevailing optimum. *Scaling* is described in [11] and *clusterized sharing* is described in [12].

Our GA also handles partial separability [11] through the use of some partial cost values, that outline some directions in which the exploration should be intensified.

B. Particle swarm optimization

Particle swarm optimization (PSO) is inspired by the observation of natural habits of bird flocking and fish schooling [13]. Candidate solutions (particles) move around in the search-space, and update their position and velocity according to simple mathematical formulae. Each particle's movement is influenced by its local best known position and is also guided toward the best known position in the search-space (Alg. 2). When improved positions are being discovered, the movements of the swarm are guided toward satisfactory solutions.

The parameters ω, ϕ_p, and ϕ_g control the behavior and efficacy of the algorithm. N is the population size.

After running a few tests with the standard PSO, it became rapidly obvious that the basic PSO is quite inefficient when applied to our problem. The main difficulty lies in the fact that the conflict resolution problem is highly constrained, in many dimensions. The PSO updates each particle using a velocity vector that may point to any direction in the search-space. As a consequence, there is a great probability that a particle that was initially conflict-free (i.e. that did not violate the separation constraints) will be moved by the position update into a region of the search-space where the

Algorithm 2 Particle swarm optimization algorithm

Initialize swarm and best known positions
while termination criterion is not met **do**
  for each particle $i = 1, \ldots, N$ in the population **do**
    for each dimension $j = 1, \ldots, n$ **do**
      Pick random numbers $r_p, r_g \in [0, 1]$
      Update the particle's velocity: $v_{i,j} \Leftarrow \omega v_{i,j} + \phi_p r_p(p_{i,j} - x_{i,j}) + \phi_g r_g(g_j - x_{i,j})$
    end for
    Update the particle's position
    Update the particle's best known position
    Update the swarm's best known position
  end for
end while
Return swarm's best known position

Algorithm 3 Differential evolution algorithm

Randomly initialize each individual
Evaluate each individual
while termination criterion is not met **do**
  for each individual $i = 1, \ldots, N$ in the population **do**
    Pick individuals $\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$ from the population
    Compute new vector $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,n})^T$ as follows:
    for each dimension $j \in \{1, \ldots, n\}$ **do**
      Pick random number $r_j \in [0, 1]$
      if $j = R$ **or** $r_j < CR$ **then**
        $y_{i,j} \Leftarrow x_{a,j} + F \times (x_{b,j} - x_{c,j})$
      else
        $y_{i,j} \Leftarrow x_{i,j}$
      end if
    end for
    Replace $\mathbf{x}$ with $\mathbf{y}$ if $\mathbf{y}$ improves the cost value
  end for
end while
Return the best individual(s) of the population

constraints are violated, possibly in several, or even all dimensions.

In order to address this issue, the original PSO algorithm has been modified as follows. When a particle violates the constraints, the position update is made by selecting a random number of dimensions, with a probability corresponding to the normalized constraint violation of each aircraft. The PSO update is then made only in these dimensions. This way, the x_i variables contributing most to the constraint violation have a greater probability to be modified than those contributing less. Another modification consists in cyclically re-initializing the swarm particles, with the hope to avoid getting trapped in a local optimum.

In our experiments, the swarm size N is set to 50 particles, ϕ_p and ϕ_g are set to 1.49618 (according to [14]), and the inertial weight ω is a decreasing function of the number of iterations since the last swarm re-initialization: $\omega = 0.5 + \frac{1}{2\times(1+\log(i-i_{restart}))}$. The first swarm re-initialization is made after 50 iterations. This number is then increased by 20 every time the swarm is re-initialized.

C. Differential evolution

Differential evolution (DE) is inspired by genetic algorithms (mutation and crossover operations) and geometric research strategies (such as the Nelder-Mead method) [16]. A single operation performing mutation and crossover is used to combine the positions of existing individuals from the population. If the new position of an individual is an improvement, it is updated in the population, otherwise the new position is simply discarded (Alg. 3).

$\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$ are chosen at random, all distinct from each other and from $\mathbf{x}_i$. R is a random index $\in \{1, \ldots, n\}$ ensuring that at least one component of $\mathbf{y}_i$ differs from this of $\mathbf{x}_i$. N is the population size, $F \in [0, 2]$ is called the differential weight and $CR \in [0, 1]$ is the crossover probability. In the following, N is set to 40, F to 0.7 and CR to 0.1.

V. Results

Tests on both problems were carried out with a horizontal separation $S_h = 5$ NM and a deviation angle $\alpha \in [-\frac{\pi}{6}, \frac{\pi}{6}]$. The algorithms were stopped after 120,000 evaluations of the cost function (CFEs).

Figures 7 to 11 show the evolution of the best individual's cost with respect to the number of CFEs for various instances of the problems. The top curves represent the overall duration of the remaining conflicts between the n aircraft. The bottom curves show the value of the cost function as soon as all conflicts have been solved.

The DE resolution of *Circle_25* is displayed on Fig. 5, and is close to the optimal roundabout configuration in which all aircraft turn right. Fig. 7 shows that GA, PSO and DE perform roughly equally, although the number of CFEs necessary to solve conflicts varies between 10,000 (DE) and 70,000 (PSO).

Table I presents average results for 20 successive runs of each algorithm on several instances of the problems. It gives the average conflict duration of the runs for which conflicts have not been solved, and the average cost value of the runs for which conflicts have been solved. The first two instances *Circle_25* and *Rand_20* are quickly solved by all 3 algorithms that handle the combinatorial

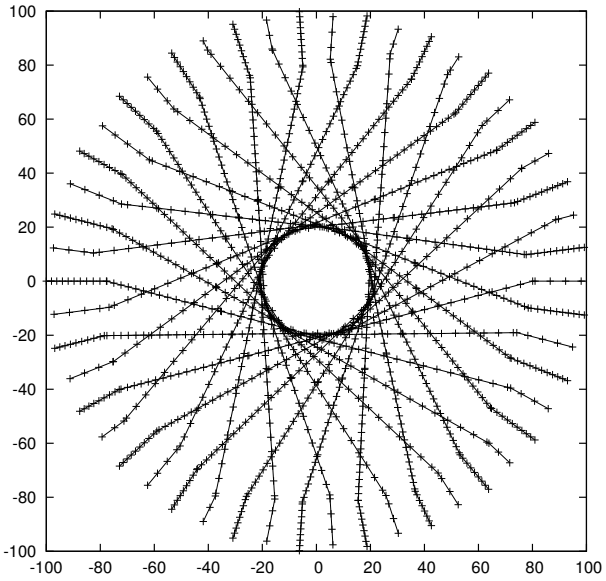


Figure 5: DE resolution of *Circle_25*

aspect on these problems well. However, some runs do not solve conflicts on large instances. While the number of unsolved runs by GA remains low as the number of aircraft increases, PSO becomes quickly inefficient to solve conflicts on larger instances.

As shown on figures 8 and 9, PSO algorithm is capable of solving the conflicts with very few iterations on small instances, whereas it struggles thereafter and remains stuck in local optima. The proposed mechanism that selectively moves particles along highly constrained directions seems to work relatively well while minimizing the constraint violations. However, for particles that do not violate the constraints, the algorithm switches back to the standard PSO update, with a velocity vector pointing to potentially any direction. Conflicts may then be generated anew. This explains the poor performances observed when trying to improve conflict-free solutions.

Figure 7: *Circle_25*: Evolution of best individual

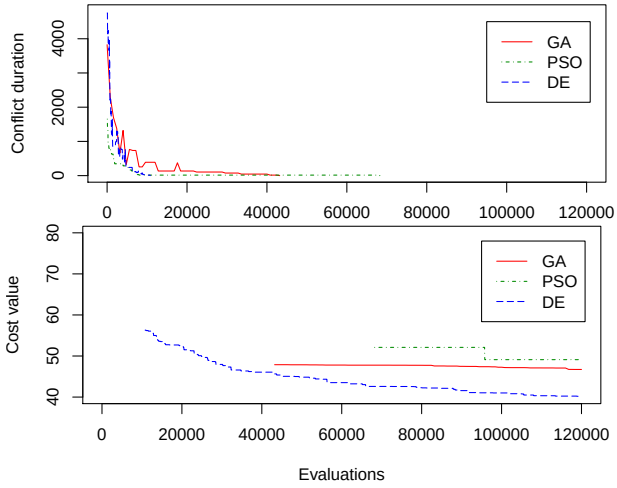


Figure 8: *Rand_20*: Evolution of best individual

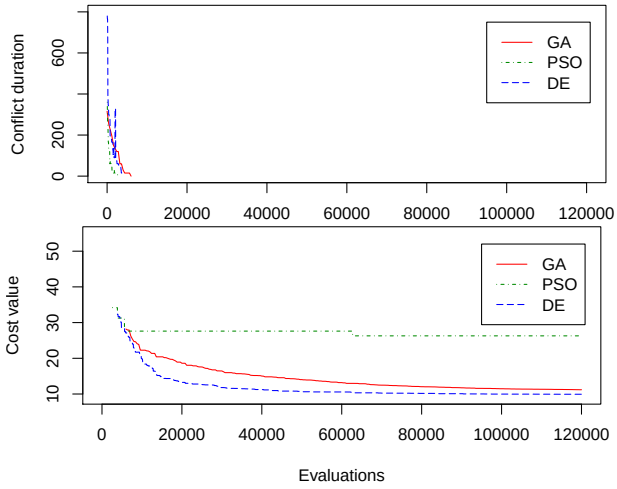
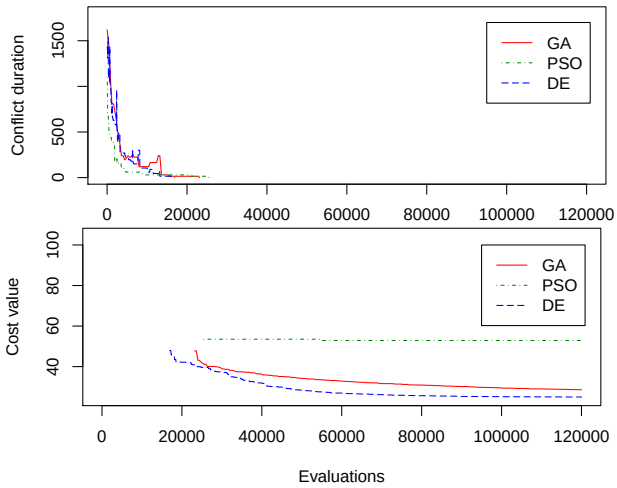
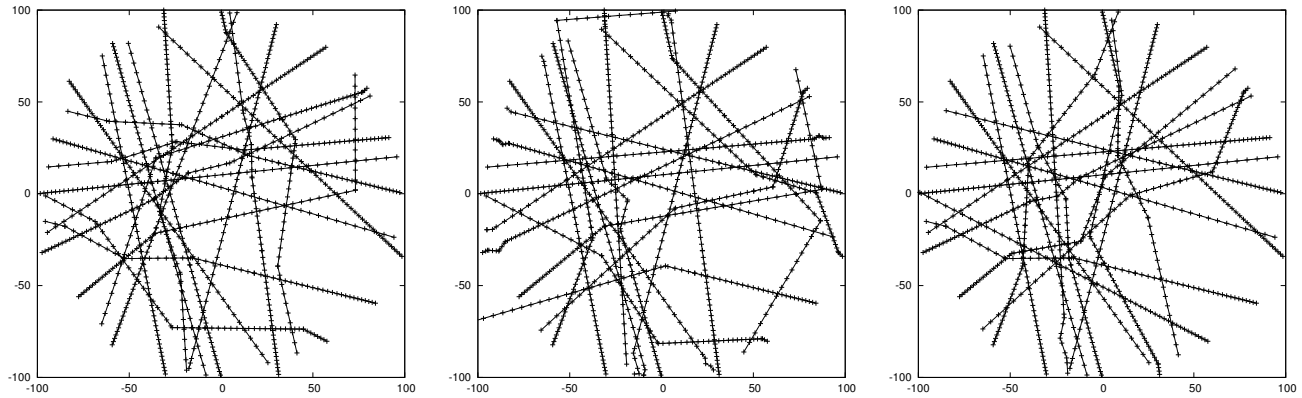


Figure 9: *Rand_30*: Evolution of best individual





(a) GA (b) PSO (c) DE

Figure 6: Resolution of *Rand_20*

Table I: Mean conflict duration and cost value over 20 runs, for 120,000 CFEs

			Conflict duration of unsolved runs				Cost of solved runs			
	Method	Unsolved runs	Min	Max	Mean	Std	Min	Max	Mean	Std
Circle_25	GA	0	-	-	-	-	46.7291	56.7154	49.5578	2.2153
	PSO	0	-	-	-	-	49.1103	55.1911	51.6155	1.3208
	DE	0	-	-	-	-	**40.1904**	54.7370	44.7435	3.9793
Rand_20	GA	0	-	-	-	-	11.1921	13.8793	12.4216	0.9086
	PSO	0	-	-	-	-	26.2831	30.2721	28.0791	1.2368
	DE	0	-	-	-	-	**9.9302**	11.8789	10.5836	0.5232
Rand_30	GA	5	15	30	21	8.2158	28.5927	36.6782	32.8227	2.1727
	PSO	6	15	15	15	0	52.8917	58.6254	55.6343	1.8290
	DE	0	-	-	-	-	**25.0167**	33.5501	29.6390	2.3471
Rand_40	GA	3	15	15	15	0	**33.7887**	46.0925	41.3232	3.1890
	PSO	12	15	105	37.5	28.2441	64.6237	76.2097	70.1448	3.6213
	DE	0	-	-	-	-	45.5938	67.4946	55.6706	6.2888
Rand_50	GA	3	30	75	45	25.9808	**56.4666**	72.5151	63.9583	3.4726
	PSO	20	30	180	129.75	41.9422	-	-	-	-
	DE	0	-	-	-	-	73.3092	115.5933	88.7457	13.0205

Figure 10: *Rand_40*: Evolution of best individual

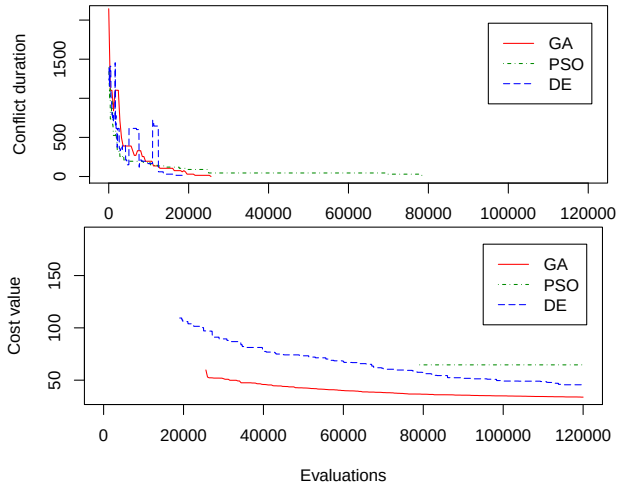
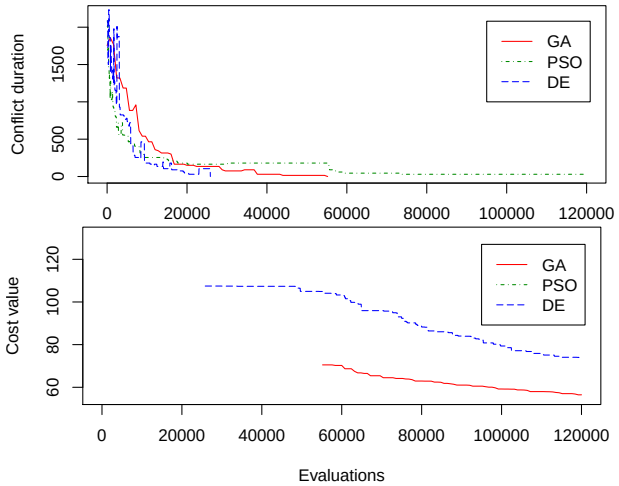


Figure 11: *Rand_50*: Evolution of best individual



VI. Conclusion and perspectives

Comparing the three methods (GA, PSO, DE) on a benchmark of traffic situations gives us an insight on the specific difficulties of the conflict resolution problem, formalized as a global optimization problem with trajectory separation constraints. This problem is combinatorial, highly constrained in many (or even all) dimensions of the search-space.

As a consequence, methods exploring the search-space by moving around in any direction are more likely to violate constraints simultaneously in many dimensions. This is clearly the case of the Particle Swarm Optimization algorithm (PSO). Methods operating selective moves in each dimension, depending on the constraint violation and cost evaluation of each flight, are clearly more efficient. This is the case of the Differential Evolution (DE) method and the Genetic Algorithm (GA) with an adapted crossover operator, that both proved more performant than the PSO algorithm. In future work, we might try PSO variants such as CPSO ([15]) that are specifically designed to explore sub-spaces of the search-space, and should perform better than the standard PSO on this problem. Other strategies may also be tried, such as sequential methods (1-against-n resolution method).

In addition, the results show that operating in the continuous space is a valid alternative to the aforementioned approaches. The chosen modeling allows to express proposed maneuvers directly into a sequence of geographical points that can be used by FMS.

In future works, we shall extend our benchmark to models involving vertical and speed maneuvers, in addition to the lateral maneuvers that were considered in the present paper. Uncertainties on the future positions and speeds of the aircraft should also be considered.

References

[1] Karim Zeghal. A Comparison of Different Approaches based on Force Fields for Coordination among Multiple Mobiles Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, Victoria, B.C., Canada, 1998.

[2] N. Durand, J-M. Alliot and Geraud Granger. FACES: a Free flight Autonomous and Coordinated Embarked Solver. ATC Quarterly, 2000.

[3] D. R. Isaacson and H. Erzberger. Design of a conflict detection algorithm for the Center/TRACON automation system. Digital Avionics Systems Conference, 1997.

[4] J-M. Alliot, J-F. Bosc, N. Durand and L. Maugis. CATS: A complete Air Traffic Simulator. 16^{th} Digital Avionics Systems Conference, 1997.

[5] L. Pallottino, E. Feron, A. Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. IEEE transactions on intelligent transportation systems, vol. 3. No. 1, March 2002.

[6] S. Cafieri, P. Brisset, N. Durand. A mixed-integer optimization model for air traffic deconfliction. Proceedings of the Toulouse Global Optimization Workshop (TOGO 2010), pp. 27-30, Toulouse, September 2010.

[7] N. Durand, J-M. Alliot, J. Noailles. Automatic aircraft conflict resolution using Genetic Algorithms. Proceedings of the Symposium on Applied Computing, Philadelphia, 1996.

[8] J-B. Gotteland, N. Durand and J-M. Alliot. Genetic Algorithms Applied to Airport Ground Traffic Optimization. Proceedings of the Congress on Evolutionnary Computation, Canberra, 2003.

[9] N. Durand, J-M. Alliot. Ant Colony Optimization for Air Traffic Conflict Resolution. 8th USA/Europe Air Traffic Management Research and Development Seminar, 2009.

[10] J. H. Holland. Adaptation In Natural And Artificial Systems, University of Michigan Press, 1975.

[11] N. Durand. Optimisation de trajectoires pour la résolution de conflits. PhD thesis, Institut National Polytechnique de Toulouse, 1996.

[12] X. Yin and N. Germay. A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization. Proceedings of the Artificial Neural Nets and Genetic Algorithms Conference, 1993.

[13] J. Kennedy and R. Eberhart. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942-1948, 1995.

[14] M. Clerc and J. Kennedy. The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, February 2002.

[15] F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimisation. IEEE Transactions on Evolutionary Computation, pp. 225-239, June 2004.

[16] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, pp. 341-359, 1997.