# Memory Based Evolutionary Algorithm for Dynamic Aircraft Conflict Resolution

Sarah Degaugue, Nicolas Durand, Jean-Baptiste Gotteland

# Memory Based Evolutionary Algorithm for Dynamic Aircraft Conflict Resolution

Sarah Degaugue[1,2], Nicolas Durand[1], and Jean-Baptiste Gotteland[1]

[1] Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France
sarah.degaugue, nicolas.durand, jean-baptiste.gotteland @enac.fr
[2] DGAC-DSNA-DTI, France

**Abstract.** In this article, we focus on a dynamic aircraft conflict resolution problem. The objective of an algorithm dedicated to dynamic problems shifts from finding the global optimum to detecting changes and monitoring the evolution of the optima over time. In the air traffic control domain, there is added value in dealing quickly with the dynamic nature of the environment and providing the controller with solutions that are stable over time. In this article, we compare two approaches of an evolutionary algorithm for the management of aircraft in a control sector at a given flight level: one is naive, i.e. the resolution of the current situation is reset to zero at each time step, and the other is memory-based, where the last population of the optimisation is stored to initiate the resolution at the next time step. Both approaches are evaluated with basic and optimised operators and settings. The results are in favour of the optimised version with explicit memory, where conflict-free solutions are found quicker and the solutions are more stable over time. Furthermore in the case of an external action, although the diversity of the population could be lower with the memory-based approach, the presence of memory does not appear to be a hindrance and, on average, improves the solver's responsiveness.

**Keywords:** Evolutionary Algorithm · Dynamic Optimisation Problem · Aircraft Conflict Resolution

## 1 Introduction

Aircraft conflict resolution is operated by air traffic controllers based on a two-dimensional representation of aircraft on a screen. The underlying problem has been modelled in many different ways allowing various metaheuristics to give efficient solutions, such as Evolutionary Algorithm (EA) [11], Ant Colony Optimisation [10], Particle Swarm Optimisation or Differential Evolution [28]. Mathematical models were also used to address this problem. In such models, the hypotheses made on trajectory predictions were generally very restrictive in order to allow mathematical resolution. For example, Pallottino's approach [22] used Mixed Integer Linear Programming and relied on constant speed trajectories that are changed all at once. This is also the case in more recent papers

from Vela, Escudero or Rey [29, 4, 24]. Constraint Programming methods [2] and Maximum Clique Search in a graph [18] have also been explored in the last decade, they can handle realistic models and perform well on small instances on which the optimality can be proved. The most realistic models take into account uncertainties and operational constraints and are thus good candidates for metaheuristics, because the trajectories need to be simulated to evaluate a set of manoeuvres. Furthermore population based metaheuristics have the great advantage to return a population of solutions instead of a single option. This gives an opportunity to imagine an intelligent support tool for air traffic controllers who could pick manoeuvre options in a pool of good solutions. Most of the research on aircraft conflict resolution studied the static problem without taking into account its dynamic over time. Indeed, aircraft constantly move in a control sector. The conflict resolution problem is thus not static but must take into account its dynamic aspect. Some changes can be modelled as continuous such as the aircraft positions or the trajectory prediction evolution over time. Other changes are discrete, such as the entrance of an aircraft in the control sector or the exit of an aircraft from the sector. An air traffic controller can suddenly decide to manoeuvre an aircraft. Because of these changes over time, conflict resolution is a dynamic problem. Besides eliminating all conflicts, the aim of conflict resolution is also to minimise delays and the number of aircraft manoeuvred. For this reason, conflict resolution is a dynamic optimisation problem (DOP).

A first approach [8] studied the operational benefits of using a memory-based EA (with basic crossover and mutation operators) and its impact on an air traffic control point of view. This article focuses on the behaviour of an EA in such a dynamic environment and introduces an optimised algorithm modifying the crossover and mutation operators. We compare the basic and optimised versions with or without a memory process on different levels of traffic densities. We also address in this article the robustness of an automatic solver with external actions on aircraft.

In part 2, we discuss the state of the art on the use of memory in EAs for DOPs. Part 3 introduces the problem modelling. Part 4 details two algorithmic adaptations of an EA for our problem. In part 5, two versions of the memory management are described. Part 6 applies the different algorithmic versions on three levels of traffic densities and discusses the effect of external actions on the dynamic simulations.

## 2 EA for Dynamic Optimisation Problems

DOPs are most commonly described in two different ways: either by a succession of static problems ([30], [5], [21]), or by a mathematical expression with time-dependant parameters ([6],[31]). The simple search for an optimal solution is no longer sufficient in a Evolutionary Dynamic Problem (EDP). Detecting changes in the environment of the current problem and consequently in the objective function are important points in the solution search.

### 2.1 Naive approach

The most naive approach to solve Dynamic Optimisation Problems (DOPs) is to reset the evolutionary process when a change occurs. A similar approach restarts the population based on the convergence of EAs [17]. Hu and Eberhart describe in [16] Rerandommisation PSO (RPSO) where all or part of the swarm is randomly moved around in the search space when a change occurs. The shortcoming of these approaches is that the past evolutionary material is not always used, although this could accelerate convergence.

### 2.2 Implicit memory

In implicit memory approaches, a local memory is added to each chromosome (e.g. adding characteristics specific to genes (redundant, recessive, etc.)). These characteristics can be used, for example, to re-introduce past good solutions. Diploid scheme [20] and triallelic scheme [13] have been introduced, respectively, by Ng and Wong in 1995 and by Goldberg and Smith in 1987. The specific nature of our problem, dealing with constantly moving aircraft, aircraft entrances and exits, makes the use of an explicit memory more adapted for this problem.

### 2.3 Explicit memory

Explicit memory approaches save information, either in the structure of individuals or in a memory external to the population, to preserve old elements of the population. Several uses of this memory are possible. Unlike implicit memory, the use of explicit memory is more controlled because it keeps information on when and which memory elements are reused. A first intuitive approach was introduced by Louis and Xu in 1996 [19] and consists of reusing old population elements when a change of environment occurs, while initialising certain chromosomes. In the same vein, in [26] authors introduce a short-term memory of ancestors present in previous generations, and in [27] authors add some of their ancestors locally to chromosomes when they are assessed as good. However, the effectiveness of these approaches is highly problem-dependent. Limits appear particularly for significant changes in the environment, as in the classic task planning problem where adding or removing a task considerably modifies the location of the optimum.

In [23], good population elements and their associated environment are stored periodically, then re-inserted when the environment changes. Instead of storing good whole individuals, and potentially their associated environment, it is also possible to store an abstract form of these individuals by recording their position in the search space. To do this, they first need to partition the search space and then define a representation matrix for this space. Following this, a spatial distribution of good individuals is obtained to guide the initialisation of a future optimisation if a change occurs [25]. This approach seems to be more useful for chaotic changes in the environment, whereas the crude safeguarding of individuals appears to be more effective for regular or cyclical changes.

## 3 Problem Modelling

This section introduces the environment and the decision variables of our problem.

### 3.1 Environment

Let us consider a situation with $n\,(\in \mathbb{N})$ aircraft flying in an en-route sector. The required separation between two aircraft is 5 nautical miles (NM) in the horizontal plane or 1000 feet in the vertical plane. Two aircraft are in conflict when there is a loss of separation in their predicted trajectories. Let us suppose that all aircraft fly with constant speeds at the same flight level. Adding different flight levels generally eases the problem because it gives more options to solve conflicts. It also partitions the problem in different sub-problems that can often be solved independently. For the sake of simplicity, we do not investigate this aspect in this article. In order to comply with air traffic controllers behaviour, our model takes into account uncertainties in the trajectory prediction. Many realistic uncertainty models have been presented in previous work [1, 3]. Here we use a simplified version of uncertainty described in [12] that increases the size of the horizontal separation standard linearly with time in the prediction. With such an uncertainty model, when looking too far ahead, many conflicts are predicted but may never occur. In order to limit this phenomenon, we limit the uncertainty growth to a time window: if $t$ is the current time, uncertainties will grow according to our model in the time window $[t, t+T]$ ($T = 6$ minutes in the experiments) and will then be capped at their value at time $t+T$ for subsequent predictions. This helps focusing on short term conflicts while keeping long term detection.

### 3.2 Decision variables

Our model considers heading change manoeuvres to solve conflicts. Here, a manoeuvre is a $\alpha$ degree heading change, starting at time $t_0$ and ending at time $t_1$. $\alpha$ is relative to the current heading. Once a manoeuvre is finished, the aircraft heads toward its destination (see figure 1).

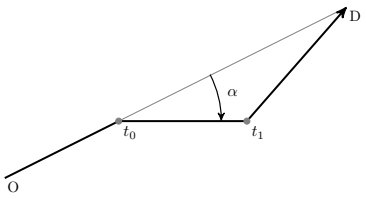


**Fig. 1.** Manoeuvre model for an aircraft flying from $O$ to $D$.

For each aircraft $i$, $(t_{0_i}, t_{1_i}, \alpha_i)$ are bounded because of aerodynamic, sector boundaries or time constraints. For instance, $t_{0_i}$ can take values in $[\underline{t_{0_i}}, \overline{t_{0_i}}]$. $\alpha_i$ is discretised with a step of 5 degrees in the range $[-45°, 45°]$, which corresponds to air traffic controllers practices.

Air traffic controllers only give one manoeuvre at a time to each aircraft. Once the manoeuvre has started, only $t_1$ can be modified (delayed or advanced to the current time plus 60 seconds (S) at the earliest). In our model, a manoeuvred aircraft can be manoeuvred again only once it is heading back to its destination (between $t_1$ and $D$ on figure 1) but a second manoeuvre cannot be predicted before the aircraft has finished its current one. This model complies to air traffic controllers habits and favours a better balance of manoeuvres between aircraft.

## 4 Algorithm versions

As first described by [15], the principle of an EA can be summarised as follows: given an evaluation function (fitness) to maximise, we initially randomly create a population of candidate solutions, and apply the fitness function as a measure of quality. At each generation, a selection is performed on the population, followed by crossovers and mutations between some individuals leading to a new population composed of some good old elements and new ones. This process is iterated until a good enough solution is found or a time limit is reached.

Two versions (basic BV, and optimised OV) of the algorithm are introduced in this section. Basic BV uses the operators introduced in [8]. In the OV version, we optimised the algorithm by defining new crossover and mutation operators, favouring diversity and by applying a local optimisation at the end of the algorithm in order to reduce as much as possible the current manoeuvres.

### 4.1 Population element structure

For a traffic situation including $n$ aircraft, a population element $e$ is a potential solution of the problem composed of $n$ genes where each gene $i$ $(g_i^e)$ is the $i^{th}$ aircraft manoeuvre $(t_{0_i}^e, t_{1_i}^e, \alpha_i^e)$ with:

- $t_{0_i}^e$, the manoeuvre start time chosen for aircraft $i$;
- $t_{1_i}^e$, the manoeuvre ending time chosen for aircraft $i$;
- $\alpha_i^e$, the deviation angle chosen for aircraft $i$. If $\alpha_i^e = 0$, the aircraft $i$ is not manoeuvred.

### 4.2 Population element fitness

In this work, we have several metrics that can be sorted by priority:

1. Solve all the conflicts;
2. Minimise the number of manoeuvres;
3. Minimise the delay due to the manoeuvres;

4. Start the manoeuvres as late as possible in order to avoid useless manoeuvres, given that detected conflicts can disappear over time with uncertainty reduction.

As these different metrics are sorted, and given that the EA will provide us with different solutions of the problem, we decided to stay in a mono-objective definition of the problem, by defining a single fitness balancing the different criteria.

Let us introduce for gene $i$ of a population element $e$:

- $d_i^e$ the delay of the aircraft $i$ due to the manoeuvre;
- $l_i^e = \overline{t_{0_i}} - t_{0_i}^e$ where $\overline{t_{0_i}}$ is the maximum time allowed to start a manoeuvre. $l_i^e = 0$ if the aircraft $i$ is not manoeuvred;
- $S^e$ the set of remaining conflicts in the population element $e$;
- $S_i^e$ the set of remaining conflicts involving aircraft $i$ in element $e$;
- $d_c$ the conflict duration for a conflict $c \in S^e$.

We introduce a local fitness as presented by Durand et al. in [9]. Each gene $i$ of a population element $e$ has a local fitness $f_i^e$, which is useful to improve the crossover and mutation steps. $f_i^e$ is expressed as follows:

$$f_i^e = \begin{cases} \frac{1}{1+\sum\limits_{c \in S_i^e} d_c} & \text{if } S^e \neq \emptyset \\ 1 + \frac{1}{1+2\times d_i^e + l_i^e} & \text{else} \end{cases}$$

Let us define $ts_c$ the starting time of conflict $c \in S^e$. If a conflict remains, it should start as late as possible. The global fitness of a population element $e$ is defined as follows:

$$F^e = \begin{cases} \frac{1}{2} - \frac{1}{2\,(1+\min\limits_{c \in S^e} ts_c)} + \frac{1}{2\,(1+\sum\limits_{c \in S^e} d_c)} & \text{if } S^e \neq \emptyset \\ \frac{1}{n} \sum\limits_{i=0}^{n} f_i^e & \text{else} \end{cases}$$

### 4.3 EA operators

**Crossover:** The crossover operator creates two new elements from two parent elements. In the BV version, one is created by mixing the genes of the two parents and the other by an arithmetic operation on the genes of the two parents. Let us consider two population elements $e_1$ and $e_2$.

For the first child $e_a$, we use the strategy described in [9] which consists in using the partial separability of a population element's fitness. For all $i \in [|1, n|]$, if $f_i^{e_1} > f_i^{e_2}$ then $g_i^{e_a} = g_i^{e_1}$ else $g_i^{e_a} = g_i^{e_2}$.

The second child $e_b$ is created by applying barycentres. For all $i \in [|1, n|]$, let us choose randomly $\lambda_{t_{0_i}}, \lambda_{t_{1_i}}, \lambda_{\alpha_i} \in [-50, 100]$ and define:

$$\begin{aligned} t_{0_i}^{e_b} &= (\lambda_{t_{0_i}} \times t_{0_i}^{e_1} + (100 - \lambda_{t_{0_i}}) \times t_{0_i}^{e_2}) \div 100 \\ t_{1_i}^{e_b} &= (\lambda_{t_{1_i}} \times t_{1_i}^{e_1} + (100 - \lambda_{t_{1_i}}) \times t_{1_i}^{e_2}) \div 100 \\ \alpha_i^{e_b} &= (\lambda_{\alpha_i} \times \alpha_i^{e_1} + (100 - \lambda_{\alpha_i}) \times \alpha_i^{e_2}) \div 100 \end{aligned}$$

The OV version emphasises the diversification role of the crossover operator by randomly creating two new random population elements one time out of three.

In both BV and OV versions, the population crossover rate $p_{cross} = 30\%$ was chosen following a former dedicated study [7].

**Mutation:** The mutation operator locally modifies one of the genes of a population element. The gene is chosen according to the value of its local fitness in order to focus on the worst values. A roulette wheel draw is performed on the set $M$ of genes that can still be modified. Let us consider a population element $e$. If $S^e \neq \emptyset$, the probability for gene $i \in M$ to be chosen is proportional to $1/min(1, f_i^e)$. When no conflict remains the probability for gene $i \in M$ to be chosen is proportional to $1/(f_i^e - 1)$.

Once a gene $i$ has been selected, we randomly modify $t^e_{0_i}, t^e_{1_i}$ or $\alpha_i^e$ if the aircraft does not yet have a manoeuvre in progress, and $t^e_{1_i}$ if the aircraft is already manoeuvred and can still be adjusted.

In the BV algorithm, we randomly change one of the three parameters of the manoeuvre of aircraft $i$. In the OV algorithm, if the current element $e$ has no conflict, we first try to remove the manoeuvre of aircraft $i$ (set $\alpha_i^e = 0$). If it creates a conflict we apply the former process.

In both BV and OV versions, the population mutation rate $p_{mut} = 40\%$ was chosen following a former dedicated study [7].

### 4.4 Sharing process

Keeping a diverse population is essential in a dynamic environment, and also very useful to avoid premature local convergence of the EA. We use a cluster based sharing process as described in [32] and [11].

The sharing process requires to define a distance between population elements. Therefore we consider three different manoeuvre directions (turn right, turn left or do not turn) in order to match the way air traffic controllers understand different resolutions. We define the distance between two population elements as the number of aircraft that are not manoeuvred in the same direction. Two population elements are zero-distant if all of their aircraft are manoeuvred in the same direction and thus belong to the same cluster. This defines the notion of cluster, grouping all the individuals giving the same manoeuvre directions to all their aircraft. There can be potentially up to $3^n$ clusters. In the experiments, considering the exponential growth of this number, and the constraints on manoeuvres in progress, we will rarely find the maximum number of clusters in a population. The sharing process helps control the diversity of the population. Let $C$ be the set of all clusters and $f_c$ the fitness of the best individual in a cluster $c \in C$. Let $c_{best}$ be the cluster in which the best individual in the population is found and $f_{best}$ its fitness. Our sharing process works in two steps. First, we define a sharing rate $s_r \in [0; 1]$. The best elements of all the clusters $c$ which respect equation 1 are automatically selected in the new population.

$$f_c \geq s_r \times f_{best} \qquad (1) \qquad\qquad F_s^e = \frac{F^e}{card(c)} \qquad (2)$$

Adjusting $s_r$ is challenging. When $s_r$ is high, only the best clusters are protected in the selection process, whereas when $s_r$ is low, the algorithm is more conservative.

The second step applies the selection process described in 4.5 to modified fitness of the population elements. The modified fitness of an element of a cluster $c$ is given by equation 2: the fitness of a population element is divided by the cardinal of its cluster.

### 4.5 Selection

The selection step is based on the stochastic remainder without replacement method [14], taking into account the fitness $F_s$ calculated during sharing.

### 4.6 Population size and Ending criterion

After a large number of experiments to check the quality of the convergence of the EA (as explained in [7]), we selected the following parameters: the population size is fixed to 200 and the algorithm is stopped after 200 generations, or when the best fitness corresponds to a non-conflicting solution and has not been improved during the last 20 generations. The EA can thus converge before reaching 200 generations.

### 4.7 Final optimisation

Some useless non-zero manoeuvres can appear and survive with the previous operators (especially the crossover and sharing operators which preserve a large population diversity). To eliminate these non-zero manoeuvres in the last generation, a final local optimisation is performed in the OV version on all individuals. $|\alpha|$ and then $t_1$ are decreased as much as possible (without creating new conflicts).

## 5 Memory Management

The traffic continuously evolves over time. Aircraft enter, leave and fly through the sector. The goal is to avoid conflicts while minimising aircraft deviations. At each time step (every 30 seconds in the experiments), every manoeuvre started or starting in less than 1 minute is updated or applied. Manoeuvres starting in more than 1 minute are ignored because they can be recalculated later.

### 5.1 Naive approach (NA)

In a classical EA initialisation, each population element is randomly created. This strategy is used in the naive approach, NA in the following. The solver only receives information about the current environment.

### 5.2 Explicit memory approach (EMA)

The explicit memory approach stores all the population elements of the last generation and reintroduces them to initialise the next resolution. Some previous solutions (respecting decision variable bounds at the previous step) may no longer be correct. An aircraft may have been manoeuvred and cannot change direction anymore, another may have finished a manoeuvre and is free to start a new one. Furthermore, the number of decision variables may have changed if some aircraft have left or entered the sector. In these cases, the genes representing the outgoing aircraft are deleted from the population elements and new genes corresponding to incoming aircraft are randomly created.

### 5.3 Summary of the tested algorithms

In the experiments, we call BVNA and BVEMA the EA versions associated with basic operators (crossover and mutation), and executed with either the NA or EMA approaches. Similarly, we call OVNA and OVEMA the EA versions associated with adapted operators, final optimisation and executed with either NA or EMA. $s_r$ was experimentally adjusted at $s_r = 0.1$, a low value favouring a large diversity in the population.

## 6 Experimental Results

### 6.1 Exercises

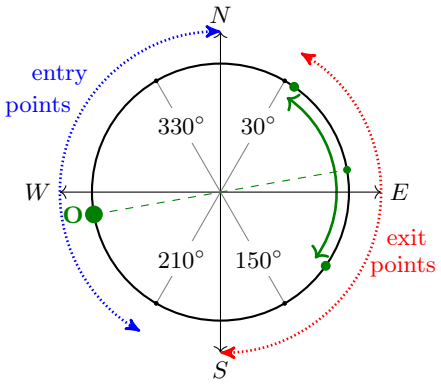


**Fig. 2.** Geometry of conflict scenario generation.

**Creation:** We create 3 scenario sizes, involving on average 35 aircraft (Baseline), 50 aircraft (Baseline+50%) and 70 aircraft (Baseline+100%) for around one hour of simulation. The Baseline scenario is already much denser than the current traffic encountered by air traffic controllers in real life. They can generally handle up to 30 aircraft, but on several independent flight levels. Each aircraft is assigned a random speed between 385 kts and 550 kts, a random entry point $O$ taken on a circle of 90 NM radius on the angle $a_{in} \in [210°; 360°]$ (see figure 2), and an exit point $D$ on the angle $a_{out} = a_{in} - 180° \pm Random(45°)$. We impose $a_{out} \in [30°; 180°]$ to avoid interactions between entering and exiting aircraft. Air traffic control generally uses an analogue semi-circular rule for the same purpose.

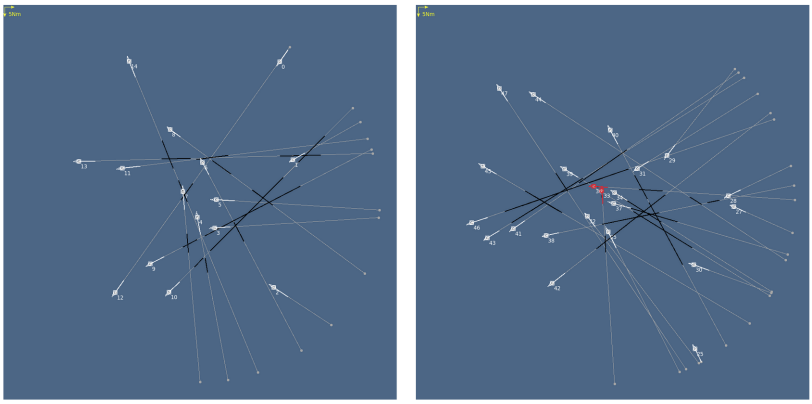

**Fig. 3.** Extracted situations from the Baseline (left) and the Baseline+100% (right) scenarios.

**Preliminary analysis:** Figure 3 shows two extracted situations simulated without resolution of the Baseline, on the left, and the Baseline+100%, on the right. The aircraft are represented by the white squares, their velocity vectors by the white lines and their previous positions by comets. The grey lines show the remaining trajectories and potential conflict zones are coloured in black. Current conflicts are coloured in red. A higher traffic density increases the number of potential conflicts.

### 6.2 First results

In a first experiment, we evaluate the combinations of the EA versions with and without memory (BVNA, BVEMA, OVNA, OVEMA) on the three exercises previously described. Each combination is run twenty times on each exercise, using different random seeds to ensure the statistical validity of the results. We

apply a Wilcoxon Rank Sum Test with continuity correction using R to compare the results between two different versions. The returned statistical elements are denoted by W and p.

**Algorithm evaluation:** The different simulations are compared using the following criteria:

- $F$: the fitness of the best element found at each resolution time step;
- $|C|$: the number of clusters at the end of each resolution;
- $|C^0|$: the number of clusters without conflict at the end of each resolution;
- $G_{tot}$: the total number of generations for each resolution;
- $G_c$: the minimal number of generations before a non-conflicting solution is found, at each resolution;
- $|S|$: the number of remaining conflicts at the end of the exercise.

Ideally, the higher $F$, $|C|$ and $|C^0|$ are, and the lower $G_{tot}$, $G_c$ and $|S|$ remain, the better the EA behaves.

| Version | BVNA | BVEMA | OVNA | OVEMA |
|---|---|---|---|---|
| Criteria | **Baseline** | | | |
| $\mu(F)$ | 1.77 | 1.79 | 1.78 | 1.79 |
| $\mu(\|C\|)$ | 84.7 | 82.1 | 120 | 120 |
| $\mu(\|C^0\|)$ | 59.9 | 54.2 | 64.0 | 59.8 |
| $\mu(G_{tot})$ | 131 | 51.9 | 129 | 51.0 |
| $\mu(G_c)$ | 3.04 | 0.04 | 3.19 | 0.05 |
| $\sum \|S\|$ | 0 | 0 | 0 | 0 |
| Criteria | **Baseline+50%** | | | |
| $\mu(F)$ | 1.76 | 1.72 | 1.77 | 1.76 |
| $\mu(\|C\|)$ | 80.8 | 79.2 | 112 | 112 |
| $\mu(\|C^0\|)$ | 55.8 | 49.8 | 55.1 | 50.4 |
| $\mu(G_{tot})$ | 135 | 66.7 | 136 | 61.1 |
| $\mu(G_c)$ | 4.86 | 0.71 | 4.22 | 0.68 |
| $\sum \|S\|$ | 0 | 0 | 0 | 0 |
| Criteria | **Baseline+100%** | | | |
| $\mu(F)$ | 1.63 | 1.59 | 1.65 | 1.66 |
| $\mu(\|C\|)$ | 82.0 | 82.9 | 115 | 115 |
| $\mu(\|C^0\|)$ | 51.0 | 45.1 | 53.1 | 50.4 |
| $\mu(G_{tot})$ | 167 | 100 | 167 | 86.2 |
| $\mu(G_c)$ | 18.6 | 19.3 | 20.8 | 2.95 |
| $\sum \|S\|$ | 0 | 4 | 0 | 0 |

**Table 1.** Algorithmic results

Table 1 shows for the three traffic densities, and the four combinations of the algorithm (BVNA, BVEMA, OVNA, OVEMA) the mean values on the 20 runs of the mean values of criteria $F$, $|C|$, $|C^0|$, $G_{tot}$ and $G_c$ on all the time steps of

the simulation where a manoeuvre occurs. The sum of remaining conflicts over the 20 runs of each exercise is shown in the last row $\sum |S|$.

The criteria between BVNA and OVNA, and between BVEMA and OVEMA improve in a majority of cases. For example in the Baseline+100% exercise, all criteria improve between BVEMA and OVEMA ($\mu(F) : 1.59 \rightarrow 1.66$ (W $> 10^7$, p $< 10^{-3}$), $\mu(|C|) : 82.9 \rightarrow 115$ (W $> 10^6$, p $< 10^{-15}$), $\mu(|C^0|) : 45.1 \rightarrow 50.4$ (W $> 10^7$, p $< 10^{-14}$), $\mu(G_{tot}) : 100 \rightarrow 86.2$ (W $> 10^7$, p $< 10^{-14}$), $\mu(G_c) : 19.3 \rightarrow 2.95$ (W $> 10^7$, p $< 10^{-13}$) and $\sum |S| : 4 \rightarrow 0$). The increase of the total number of clusters is notable. The new crossover operator creating new individuals one time out of three in the OV version could explain this phenomenon. The number of remaining conflicts drops to zero in the OV version.

Averages of $\mu(F)$ are similar for OVNA and OVEMA but differences appear when looking at the number of generation and cluster criteria. Using memory does not seem to penalise the diversity of the whole population. The constant reuse of former population elements does not impact the total number of clusters. However, even if the number of clusters $|C|$ are similar, the number of clusters without conflict remains slightly higher without memory, showing that starting with a random population remains better for the diversity of acceptable solutions. Keeping a high diversity is important in the eventuality of external actions on aircraft. The main advantage of using memory is shown with the optimised version for which $\mu(G_{tot})$ and $\mu(G_c)$ decrease a lot with the use of memory. For example, on the Baseline+100% exercise, the mean number of generations necessary to find a conflict-free solution is divided by almost ten, dropping from 20.8 to 2.95 ( W $> 10^7$, p $< 10^{-15}$). This last point could be very advantageous if the EA was used to help an air traffic controller making decision in real time.

**Aeronautical performances:** Table 2 evaluates the following criteria on the same simulations. The lower the criteria are, the better the aeronautical performances are.

- $M$: the number of manoeuvres divided by the number of aircraft at the end of each simulation;
- $D$: the average percentage of additional flight time per aircraft;
- $V$: the percentage of aircraft with varying manoeuvres planned. A manoeuvre is varying if it has at least been planned in the opposite direction (turn right then left or turn left then right) between two successive resolutions.

In the most of the cases, especially when the traffic density increases, OVNA and OVEMA have better results than BVNA and BVEMA. When memory is used, the OV version tends to minimise the number of manoeuvres performed, but the manoeuvre durations are higher. The balance between these two criteria is modified with the use of memory. This can be adjusted in the fitness definition.

The major observation here is the improved stability of the manoeuvres planned by the conflict solver when memory is used. Indeed, for the most dense exercise, criterion $V$ drops from 33% (without memory) to 9% (with memory) (W $= 0$, p $< 10^{-7}$), showing that with memory, only 9% of the aircraft are planned

| Version | BVNA | BVEMA | OVNA | OVEMA |
|---|---|---|---|---|
| Criteria | **Baseline** | | | |
| $M$ | 0.28 | 0.28 | 0.30 | 0.28 |
| $D$ | 1.06 | 1.18 | 1.18 | 1.11 |
| $V$ | 14.1 | 9.11 | 12.5 | 7.14 |
| Criteria | **Baseline+50%** | | | |
| $M$ | 0.28 | 0.32 | 0.27 | 0.27 |
| $D$ | 1.23 | 2.10 | 1.30 | 1.55 |
| $V$ | 17.7 | 7.82 | 14.1 | 6.54 |
| Criteria | **Baseline+100%** | | | |
| $M$ | 0.64 | 0.62 | 0.59 | 0.54 |
| $D$ | 3.08 | 3.91 | 2.84 | 3.55 |
| $V$ | 42.1 | 20.8 | 33.3 | 9.04 |

**Table 2.** Aeronautical results

an opposite direction between two time steps. This could be a real advantage if our EA had to help an air traffic controller make decisions over time.

### 6.3 External action impacts

In this section, we focus on the effect of external actions on EA resolutions. At some time steps (every 300 seconds if possible, or more if not), a random manoeuvre is associated with an aircraft (in the aircraft manoeuvring bounds) and both the state of the environment and the resolution at the next time step are saved. The main simulation is run using the OVEMA version. When an external manoeuvre is applied, we compare two resolutions, OVNA and OVEMA at this specific time step.

We randomly apply a manoeuvre to one aircraft that can still be manoeuvred. If this manoeuvre does not cause a conflict within the next three minutes, it is added to the current solution, otherwise the process is repeated until an acceptable manoeuvre is found. This random action is initiated if the EA has at least one conflict-free solution at the current time and if at least one aircraft can be manoeuvred.

In table 3, for each exercise, we evaluate the mean values of $F$, $|C|$, $|C^0|$, $G_{tot}$ and $G_c$ with the OVNA and OVEMA solvers. The percentage of times for which one version is strictly better than the other is shown in the brackets next to the mean value of the criteria.

The mean fitness is generally better with memory than without, and the fitness is strictly better without memory only 11%, 17% and 27% of the time. As previously, the major result concerns the number of generations necessary to optimise a solution or to find a conflict-free solution. For the most dense scenario (Baseline+100%), the total number of generations $G_{tot}$ drops from 154 to 104 ($W > 10^5$, $p < 10^{-15}$) with memory and the number of necessary generations to find a conflict-free solution drops from 39 to 21 ($W > 10^5$, $p < 10^{-15}$) with

| | **Baseline** | | **Baseline+50%** | | **Baseline+100%** | |
|---|---|---|---|---|---|---|
| Criteria | OVNA | OVEMA | OVNA | OVEMA | OVNA | OVEMA |
| $\mu(F)$ | 1.76 (11%) | 1.84 (80%) | 1.77 (17%) | 1.76 (36%) | 1.57 (27%) | 1.6 (54%) |
| $\mu(\|C\|)$ | 93 (40%) | 99 (53%) | 102 (40%) | 101 (40%) | 111 (46%) | 111 (43%) |
| $\mu(\|C^0\|)$ | 57 (48%) | 61 (42%) | 57 (53%) | 54 (30%) | 49 (57%) | 48 (26%) |
| $\mu(G_{tot})$ | 92 (5%) | 47 (54%) | 108 (24%) | 70 (56%) | 154 (10%) | 104 (64%) |
| $\mu(G_c)$ | 6 (6%) | 2.4 (21%) | 7.6 (3%) | 5.4 (33%) | 39 (5%) | 21 (61%) |

**Table 3.** Effect of external actions

memory. Using a memory approach has not prevented the EA to adjust to an unexpected event.

## 7 Conclusion

In this article, we introduce an optimised version of an EA combined with a memory process and compare it with a basic version on a very dense dynamic conflict resolution optimisation problem. We define a new crossover operator to help the EA keep a diverse population even when a memory process is used. We also define a new mutation operator to keep reduced manoeuvres allowing aircraft to remain free for future manoeuvres. We add a final optimisation process to help remove useless manoeuvres and reduce delays. We use a cluster based sharing process with a low sharing rate to make sure that the population covers the search space as much as possible through the simulations.

Using an explicit memory process drastically reduces the total number of generations and the number of generations necessary to find a conflict-free solution. It also reduces the variation of manoeuvres planned over time. These two results are essential if such an algorithm was used to help air traffic controllers make real time decisions in a dynamic environment.

We show that the memory based approach can handle external actions and still quickly find good solutions despite the fact that old options are kept in the population. This is an important point if such an algorithm should interact with an air traffic controller making decisions that are not present in the manoeuvres covered by the population.

For future research, it could be wise, at every time step, to run in parallel the EA with several random seeds and keep the best solution found. This could be tested and compared with or without memory. We could also imagine a hybrid version of the naive and explicit memory approaches, where only the best element of each cluster at the last generation would be reintroduced in the next population initialisation. The rest of the population could then be randomly created which would favour some diversity in the population.

This work will be tested with air traffic controllers to measure the capacity for collaboration between humans and a population based automatic solver and the quality of interactions with such a memory based algorithm.

## References


1. Allignol, C., Barnier, N., Durand, N., Alliot, J.M.: A new framework for solving en-routes conflicts. In: 10th USA/Europe Air Traffic Management Research and Developpment Seminar (2013)
2. Allignol, C., Barnier, N., Durand, N., Gondran, A., Wang, R.: Large Scale 3D En-Route Conflict Resolution. In: ATM Seminar, 12th USA/Europe Air Traffic Management R&D Seminar. Seattle, United States (Jun 2017), https://enac.hal.science/hal-01592235
3. Allignol, C., Barnier, N., Durand, N., Gondran, A., Wang, R.: Large Scale 3D En-Route Conflict Resolution. In: ATM Seminar, 12th USA/Europe Air Traffic Management R&D Seminar. Seattle, United States (Jun 2017), https://hal-enac.archives-ouvertes.fr/hal-01592235
4. Alonso-Ayuso, A., Escudero, L., Martin-Campo, F.: Collision avoidance in air traffic management: a mixed-integer linear optimization approach. IEEE Transactions on Intelligent Transportation Systems **12**(1), 47–57 (2011)
5. Aragon, V.S., Esquivel, S.C.: A evolutionary algorithm to track changes of optimum value locations in dynamic environments. Journal of Computer Science and Technology **4**(3), 127–134 (2004)
6. Bosman, P.A.N.: Learning and anticipation in online dynamic optimization. In: in Dynamic, E.C., Environments, U. (eds.) Studies in Computational Intelligence, vol. 51, pp. 129–152. Springer Berlin Heidelberg (2007)
7. Degaugue, S., Gotteland, J., Durand, N.: Algorithme évolutionnaire pour la résolution, en continu, de conflits aériens. In: ROADEF (2023)
8. Degaugue, S., Durand, N., Gotteland, J.B.: Impact of Explicit Memory on Dynamic Conflict Resolution. In: 10th International Conference on Research in Air Transportation (ICRAT 2022). p. paper 53. Tampa, United States (Jun 2022), https://hal.science/hal-03878000
9. Durand, N., Alliot, J.M.: Genetic crossover operator for partially separable functions. In: GP 1998, 3rd annual conference on Genetic Programming. Madison, United States (Jul 1998), https://enac.hal.science/hal-00937718
10. Durand, N., Alliot, J.M.: Ant Colony Optimization for Air Traffic Conflict Resolution . In: ATM Seminar 2009, 8th USA/Europe Air Traffic Management Research and Developpment Seminar. Napa, California, United States (Jun 2009), https://enac.hal.science/hal-01293554
11. Durand, N., Alliot, J.M., Noailles, J.: Automatic aircraft conflict resolution using genetic algorithms. In: Proceedings of the Symposium on Applied Computing, Philadelphia. ACM (1996)
12. Durand, N., Gotteland, J.B., Matton, N.: Visualizing complexities: the human limits of air traffic control. Cognition, Technology and Work (Feb 2018). https://doi.org/10.1007/s10111-018-0468-0, https://hal-enac.archives-ouvertes.fr/hal-01707751
13. Goldberg, D.E., Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: ICGA (1987)
14. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Reading MA Addison Wesley (1989)
15. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan press (1975)
16. Hu, X., Eberhart, R.: Adaptive particle swarm optimization: detection and response to dynamic systems. In: Proceedings of the 2002 Congress on Evolutionary

Computation. CEC'02 (Cat. No.02TH8600). vol. 2, pp. 1666–1670 vol.2 (2002). https://doi.org/10.1109/CEC.2002.1004492
17. Krishnakumar, K.: Micro-Genetic Algorithms For Stationary And Non-Stationary Function Optimization. In: Rodriguez, G. (ed.) Intelligent Control and Adaptive Systems. vol. 1196, pp. 289 – 296. International Society for Optics and Photonics, SPIE (1990). https://doi.org/10.1117/12.969927, https://doi.org/10.1117/12.969927
18. Lehouillier, T., Omer, J., Soumis, F., Desaulniers, G.: A flexible framework for solving the air conflict detection and resolution problem using maximum cliques in a graph (06 2015)
19. Louis, S., Xu, Z.: Genetic algorithms for open shop scheduling and re-scheduling. In: ISCA 11th Intl. Conf. on Computers and their Applications. pp. 99–102 (1996)
20. Ng, K.P., Wong, K.C.: A new diploid scheme and dominance change mechanism for non-stationary function optimization. (1995), https://cir.nii.ac.jp/crid/1373949023319151361
21. P. Rohlfshagen, P.K.L., Yao, X.: Dynamic evolutionary optimisation: An analysis offrequency and magnitude of change. In: Proceedings of the 2009 Genetic and Evolutionary Computation Conference GECCO'09. pp. 1713–1720 (2009)
22. Pallottino, L., Féron, E., Bicchi, A.: Conflict resolution problems for air traffic management systems solved with mixed integer programming. IEEE Transactions on Intelligent Transportation Systems **3**(1), 3–11 (2002)
23. Ramsey, C.L., Grefenstette, J.J.: Case-based initialization of genetic algorithms. In: Proceedings of the 5th International Conference on Genetic Algorithms. p. 84–91. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
24. Rey, D., Rapine, C., Fondacci, R., Faouzi, N.E.: Minimization of potential air conflicts through speed regulation. Transportation Research Record: Journal of the Transportation Research Board **2300**, 59–67 (2012)
25. Richter, H., Yang, S.: Memory based on abstraction for dynamic fitness functions. In: Proceedings of the 2008 Conference on Applications of Evolutionary Computing. p. 596–605. Evo'08, Springer-Verlag, Berlin, Heidelberg (2008)
26. Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 3, pp. 1843–1850 Vol. 3 (1999). https://doi.org/10.1109/CEC.1999.785498
27. Trojanowski, K., Michalewicz, Z., Xiao, J.: Adding memory to the evolutionary planner/navigator. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97). pp. 483–487 (1997). https://doi.org/10.1109/ICEC.1997.592359
28. Vanaret, C., Gianazza, D., Durand, N., Gotteland, J.B.: Benchmarking conflict resolution algorithms. In: ICRAT 2012, 5th International Conference on Research in Air Transportation. Berkeley, United States (May 2012), https://hal.science/hal-00863090, http://www.icrat.org//icrat/Author/CharlieVanaret669/FINAL-329-cfp-Vanaret.pdf
29. Vela, A., Solak, S., Singhose, W., Clarke, J.: A mixed integer program for flight-level assignment and speed control for conflict resolution. In: Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference. IEEE (2009)
30. Weicker, K.: An analysis of dynamic severity and population size. Parallel Problem Solving from Nature **VI** (2002)
31. Woldesenbet, Y.G., Yen, G.G.: Dynamic evolutionary algorithm with variable relocation. IEEE Transactions on Evolutionary Computation **13**(3), 500–513 (2009)

32. Yin, X., Germay, N.: A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In: Albrecht, R.F., Reeves, C.R., Steele, N.C. (eds.) In proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference, Innsbruck Austria. Springer-Verlag (1993)