ENRI Int. Workshop on ATM/CNS. Tokyo, Japan (EIWAC2015)

[EN-A-064] Aircraft Trajectory Planning with Dynamical Obtacles by Artificial Evolution and Convex Hull Generations

+ S. Pierre * D. Delahaye ** S. Cafieri **

* MAIAA, ENAC
French Civil Aviation University
7 Avenue Edouard Belin, 31055, Toulouse, France

* spierre@etud.insa-toulouse.fr

** [daniel.delahaye—sonia.cafieri]@enac.fr

Abstract: Air Traffic Management (ATM) ensures the safety of flights by optimizing flows and maintaining separation between aircraft. Many ATM applications involve some aircraft trajectory optimization in order to improve the performance of the overall system. Trajectories are objects belonging to spaces with infinite dimensions. Widely used approaches are based on discretization, sampling trajectories at some regular points, and then using appropriate representations to reduce the dimension of the search space. We propose an approach in which trajectories in a two-dimensional space are designed with the help of convex hull generation. By using static as well as moving obstacles for which the position and the size are controlled by artificial evolution, we propose a new algorithm for efficient trajectory planning in Terminal Maneuvering Areas.

Keywords: Trajectory Planning, TMA, Optimization, Obstacle Avoidance

1 INTRODUCTION

Trajectory planning is crucial in Air Traffic Management (ATM) to regulate air traffic flows while ensuring flight safety. Trajectories are designed so as to avoid aircraft potential conflicts as well as to optimize some criteria, such as cost index or environmental criteria (noise abatement, pollutant emission, etc.). Depending on the considered time horizon, the following kinds of trajectory planning can be carried out:

- At a strategical level, only macroscopic indicators like congestion, mean traffic complexity, delays can be taken into account, as well as the presence of obstacles;
- at a pre-tactical level, the accuracy of previous indicators, specially congestion and complexity, increases while at the same time early conflict detection can be performed;
- finally, at the tactical level, conflict resolution is the major concern and optimality of the trajectories is only marginally interesting.

In this work, we focus on path planning in Terminal Maneuvering Areas (TMA), that are the areas surrounding one or more neighbooring airports, where arriving and departure routes (also called STAR and SID respectively) have to be handled. Such a planning is done at a strategical level, and aims at designing trajectories avoiding obstacles.

Aircraft trajectories are usually designed into three steps. The first step consists in the design of the two-dimensional shape between two points, respectively the origin and the destination point. Then, an optimal altitude profile is computed in order to create a full three-dimensional shape. Finally, the speed profile is computed in order to optimize some cost criteria (fuel, cost index). In this work, we propose a new approach to optimize the two-dimensional shape of an aircraft trajectory. We consider that an aircraft systematically receives its optimal altitude and speed profiles. So, the obtained 2D shapes are supposed to be evaluated in the three-dimensional space by using some given altitude profiles.

The paper is organized as follows. Section 2 outlines the main existing approaches for trajectory planning. Section 3 presents the proposed model and algorithm. First, the trajectory shape design is introduced, then a combinatorial optimization problem and an evolutionary algorithm for its solution, used to carry out such a design, are presented. Section 4 present an extension of the proposed approach to take efficiently into account the case of dynamic obstacles. Some results validating such an approach are discussed in Section 5. Finally, Section 6 draws some conclusion.

2 STATE OF THE ART

Trajectories are mathematical objects belonging to spaces with infinite dimensions. Widely used approaches for their design are based on discretization.

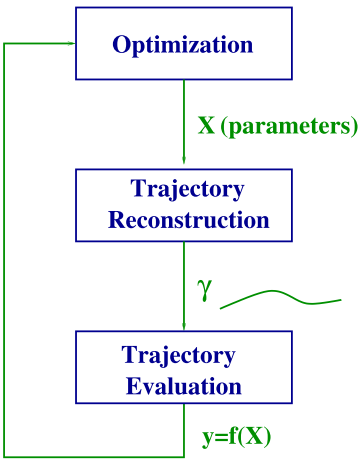

Figure 1 .The optimization process control the X vector in order to build a trajectory γ for evaluation.

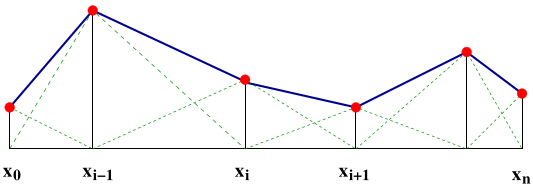

Figure 2 Piecewise linear interpolation. By summing the triangle shapes (dash lines) controlled by the positions of the red dot and the associated extension, we can generate a full linear piecewise interpolation (solid line).

The general idea is to use a discrete number of parameters describing the trajectory, optimize such parameters with respect to some selected criterion to design a given class of trajectory shapes, and finally build a trajectory γ. This process is summarized in Fig. 1. Starting from a discrete set of points, a way to build a trajectory γ is based on using interpolation. Piecewise linear interpolation is the simplest piecewise interpolation method. An example of such a linear piecewise interpolation is shown in Fig. 2. In this case, the derivative of the resulting curve is not continuous. In order to fix this drawback, one can use piecewise quadratic, piecewise cubic interpolation [8] or cubic spline interpolation [2], which ensures smooth C^2 shapes. Alternatively, when interpolating the given points is not a hard constraint, one can use some control points which control the shape of a given trajectory without forcing this trajectory to go through such points. Among such kind of approximation approaches, we may recall the ones based on Bézier curves [6] or on B-splines [3]. If many points have to be considered, using a Bézier curve one has to manipulate polynoms with high degree. B-splines allow to circumvent this weak point. A B-spline is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition. An example of B-Spline of order one is given on Fig. 3. To increase the smoothness of the resulting shape, one may use B-splines of order three. When many aircraft trajectories samples are available (for instance, from radar), one can build a dedicated basis and minimize the number of coefficients for trajectory reconstruction:

$$\gamma_i(t) = \sum_{k=1}^{k=K} a_{ik}\psi_k(t) \qquad (1)$$

For instance, Principal Component Analysis (PCA [1]) can be used to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables, called principal components.

Another easy way to build trajectory is to use some reference trajectories (regular trajectories used by aircraft) and to compute a weighted sum of such reference trajectories to build a new one. Considering two (or more) reference trajectories γ_1, γ_2 joining the same origin-destination pair (see Fig. **??**) (past flown trajectories may be considered), one can create a new trajectory γ_α by using an homotopy :

$$\gamma_\alpha = (1-\alpha)\gamma_1 + \alpha\gamma_2, \quad \alpha \in [0,1]. \qquad (2)$$

Remark that the above approaches do not take into account obstacles in the design process. Obstacles can nevertheless be included as a penalty in the objective function. Trajectory design in a constrained space (*i.e.*, with obstacles) has been looked at through various techniques. They include the A* algorithm to provide obstacles-free trajectories, followed by the use of some smoothing algorithm to improve their regularity [10]. Another approach consists in using a branch and bound algorithm where the branching strategy is associated to the way obstacles avoided, bypassing them in one direction or the other [5].

In this paper we propose an approach based on obstacle convex hulls generation in order to create paths

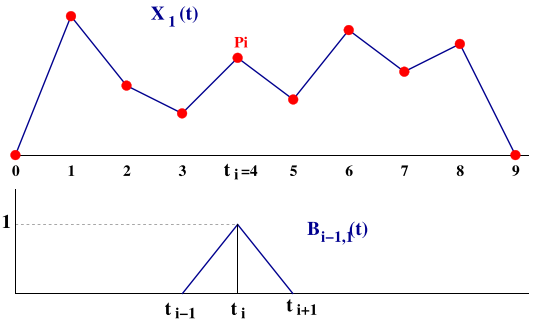

Figure 3 An example of B-spline of order 1.

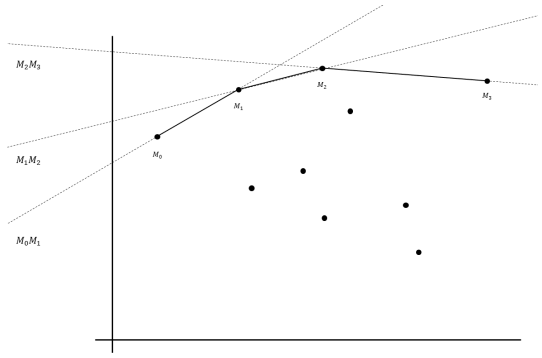

Figure 4 Jarvis March algorithm

around some given sets of obstacles. Two sets of obstacles are then considered: one of real obstacles and one of virtual obstacles. The proposed model is presented in the next section.

3 OPTIMIZATION MODEL AND ALGORITHM

3.1 Trajectory shape design

Let us consider, in a two-dimensional space, two points representing the origin and the destination of the trajectory to be designed (the origin being the point with the smallest x-axis value), and a set Ω of obstacles. Let us model obstacles in Ω by sets of points defining their 2D contour shapes. These points have x-axis values between those of the origin and destination. We suppose these points defining convex sets; if this is not the case, we consider the associated convex envelope as the obstacle instead of the original one. We also suppose that the points defining obstacles are ordered clockwise or counter-clockwise.

Building on the technique from R.A. Jarvis, known as *Jarvis March* [9], we propose a trajectory planning approach based on building convex hulls around obstacles. Specifically, we use the following adaptation of the Jarvis March or gift wrapping algorithm. Let G be a set of two-dimensional points which represent obstacles. We first consider the point M_0 such that:

$$\forall P \in G,\ (P)_x \geq (M_0)_x \tag{3}$$

where $(P)_x$ denotes the x-value of point P. We then look for the next point M_1 such that:

$$\forall P \in G,\ P \text{ is on the left side of } \overrightarrow{M_0M_1}$$

Iterating this procedure, we get the point $M_{n+1} \in G$ such that:

$$\forall P \in G,\ P \text{ is on the left side of} \overrightarrow{M_nM_{n+1}}$$

We stop once M_{n+1} is equal to M_0. The set $C = \{M_0, \ldots, M_n\}$ is the so called convex hull of G. An example of such convex hull computation is shown in Fig. 4. The convex hull created through the above technique, in the case of two obtacles, describes two paths around the obstacles like illustrated in Fig. 5. Remark that the described algorithm is not able to generate shapes like the one shown in Fig. 6. To build a trajectory with a shape like the one shown in Fig. 6, which is more close to what is done in practice in the operational context, and whose length is less then the one of a trajectory obtained with the above algorithm,

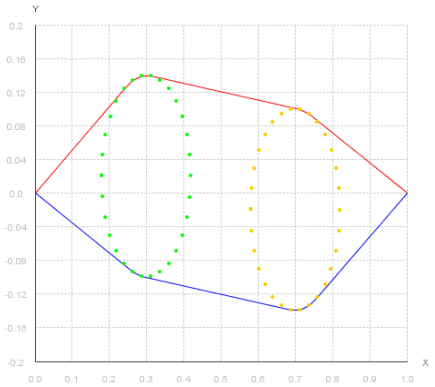

Figure 5 Trajectories around two obstacles, built by using the Jarvis March algorithm. Two segments join the extreme points.

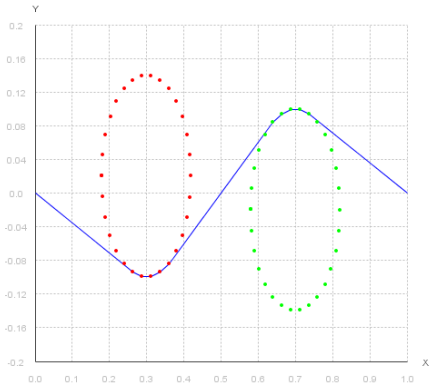

Figure 6 Trajectory around two obstacles. It is buit with two segments, one convex and another one concave.

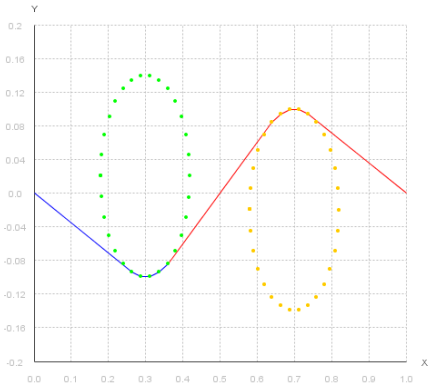

Figure 7 Trajectory around two obtacles. The new version of the Jarvis March algorithm is used.

we propose a variant of the algorithm which generates piecewise convex hull segments. To this aim, we first partition the set of obstacles into two subsets: the set O_a of obstacles to be avoided by a "convex" trajectory, and the set O_b of obstacles to be avoided with a "concave" trajectory. Let G_a (respectively G_b) be the set of points describing O_a (respectively O_b). Similarly to what is done when the Jarvis March algorithm is used, we first consider the point M_0 such that:

$$\forall P \in G_a,\ (P)_x \geq (M_0)_x \tag{4}$$

Then, proceeding iteratively, we define the point M_{n+1}:

$$\forall P \in G_a,\ P \text{ is on the left side of } \overrightarrow{M_n M_{n+1}} \tag{5}$$

The difference with the above Jarvis March algorithm lies in the stopping criterion. This time we stop the algorithm when the following condition is satisfied:

$$\forall P \in G_b,\ P \text{ is on the right side of } \overrightarrow{M_n M_{n+1}} \text{ with } M_n, M_{n+1} \in G_a \tag{6}$$

The point M_{n+1} is not included in the convex hull C_a. We get a kind of semi convex hull. An example of trajectory around two obstacles obtained by the proposed algorithm is given in Fig. 7. Note that the above algorithm only builds the blue part of the trajectory in Fig. 7. To produce the red part, the algorithm has to be applied again with O_b and another obstacle O_c. This time, since we want to bypass the obstacle through a concave trajectory, we follow exactly the same procedure except that all the $right - left\ side$ comparisons are reverted, *i.e.,* the stopping criterion is changed to:

$$\forall P \in G_c,\ P \text{ is on the left side of } \overrightarrow{M_n M_{n+1}} \text{ with } M_n, M_{n+1} \in G_b \tag{7}$$

To connect the two pieces of trajectories created as described above, we just add the last point of C_a to G_b. Finally, a full trajectory is built by considering the starting point S as the first obstacle and the ending point E as the last obstacle, corresponding to the origin and the destination. Such a trajectory is like the one in Fig. 6.

We can now describe our optimization model.

3.2 Decision variables

Let Ω be the set of all n obstacles to be avoided. Let O_p and $O_q \in \Omega$ be two obstacles defined by the set of points G_p and G_q:

$$p < q \Rightarrow \left(M_p\right)_x \leq \left(M_q\right)_x \tag{8}$$

with

$$\left(M_p\right)_x \leq (P)_x, \forall P \in G_p \tag{9}$$

and

$$\left(M_q\right)_x \leq (P)_x, \forall P \in G_q \tag{10}$$

We consider obstacles ordered from the "left to the right". The decision variables of our optimization model are binary variables defined as follows:

$$X_i = \begin{cases} 1 & \text{if obstacle } i \text{ is to be bypassed} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

and

$$Y_i = \begin{cases} 1 & \text{if obstacle } i \text{ is to be bypassed by the bottom} \\ & \text{(convex piece of trajectory)} \\ 0 & \text{if obstacle } i \text{ is to be bypassed by the top} \\ & \text{(concave piece of trajectory)} \end{cases} \tag{12}$$

The space state of the optimization problem is then defined by $2 \times n$ variables, with n the total number of obstacles. Fig. 8 gives an example of decision variables values where two obstacles have to be avoided.

3.3 Objective function

The objective function, to be minimized, is composed of two parts, each accounting for a criterion to be taken into account. The first one is linked to the length of the (discrete) trajectory T:

$$L(T) = \sum_{i=1}^{n} l(P_i, P_{i-1}) \tag{13}$$

with $T = \{P_i, i \in \{1, ..., n\}\}$ and $l(M, N)$ the distance between the two points M and N. The second one

is associated to pieces of trajectories going through obstacles: even if a piece of trajectory crosses an obstacle in the two-dimensional space, it may happen that it does not intersect the obstacle on the vertical plan. This situation can be detected when the three-dimensional space is considered, in a second step, after that the two-dimensional shape of the trajectory is obtained. We introduce the following function:

$$D(T) = \sum_{i=1}^{m} \sum_{j=1}^{n} R_i \times \Delta(P_j, O_i) \tag{14}$$

where R_i is the cost for going inside the obstacle O_i, and $\Delta(P, O) = 1$ if the point P is inside the obstacle O and $\Delta(P, O) = 0$ otherwise. To detect if a point is or not inside an obstacle, we use the assumptions that obstacles are convex and the fact the points defining obstacles are ordered in the clockwise direction. Then:

$$P \text{ is in } O_p \Leftrightarrow (P, M_1)(P, M_2) \geq 0, \forall M_1, M_2 \in G_p$$

where $P \in \mathbb{R}^2$, O_p is defined by the set of points G_p and (P, M) is the cross product between P and M.

Finally, the objective function of our optimization model is defined as follows:

$$F(T) = L(T) + D(T) \tag{15}$$

and has to be minimized.

3.4 Genetic algorithm

The problem to be solved is a combinatorial optimization problem, whose complexity is directly related to the number of obstacles. We then address the problem by using an artificial evolution algorithm, where a stochastic tournament selection process is used associated with an elitism principle. More specifically, we use a genetic algorithm, where three kinds of mutation operators, $\psi_i\, i = 1 \ldots 3$, are applied with a given probability. The first mutation operator (ψ_1) randomly generate two new vectors X and Y. It is very disruptive and is mainly applied at the beginning of the evolution process. The second operator (ψ_2) randomly change a percentage of bit in X and Y (this percentage is diminishing with the generation number). The third one (ψ_3), in the case of a trajectory crossing some obstacles, randomly chose an obstacle O_p between the ones which are crossed by the trajectory and change the values of X_p and Y_p. Crossover operators have also been developed, but from some numerical test it appears that they do not really improve the performance of the algorithm, so only mutation operators are kept.

4 MODEL EXTENSION

4.1 Adding a time dimension

4.1.1 New obstacle's definition

Aircraft trajectory design is also subject to dynamic obstacles like storms moving in the airspace which have to be avoided. We then extend the previous approach so as to take into account dynamic obstacle avoidance. We consider obstacles in three dimensions, including now the time dimension in their definition. So, we define obstacles as discrete structures composed by a starting time and an ending time, a set of points in two-dimensional space, and a speed vector. Let O_p be a three-dimensional obstacle. Let t_o be the starting time, t_f the ending time, G_o the set of points at t_o and v the speed vector. Then, O_p is defined by $O_P = \{t_o \in \mathbb{R}^+, t_f \in \mathbb{R}^+, G_o, v \in \mathbb{R}^2\}$. Let $G(t)$ define the set of points representing the obstacle at time t:

$$G(t) = \begin{cases} \emptyset & , t > t_f \\ \emptyset & , t < t_o \\ \{P + v \times t, P \in G_o\} & , t \geq t_o, t \leq t_f \end{cases} \tag{16}$$

4.1.2 Obstacles intersection

In order to dynamically detect when a generated trajectory crosses a given moving obstacle O_p, we use the pretty simple following assertion.
Let $P = (x, y, t) \in \mathbb{R}^2 \times \mathbb{R}^+$ be a point,

$$P \in O_p \Leftrightarrow (1) \text{ and } t_o \leq t \leq t_f \tag{17}$$

where (1) is:

$$(P, M_1(t))(P, M_2(t)) \geq 0,\ \forall M_1(t), M_2(t) \in G_p(t). \tag{18}$$

This corresponds to the spatial-time extension of the obstacles.

	X	Y
Obstacle 1	1	1
Obstacle 2	1	0

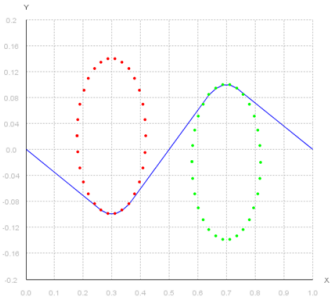

Figure 8 In this example both obstacles are active, the second one has to be avoided by the top.

Table 1 Dimension of the state space

Time step	Ω	Ω_m
0.5	5.904×10^4	3.874×10^8
0.2	5.904×10^4	2.058×10^{14}
0.1	5.904×10^4	7.178×10^{23}

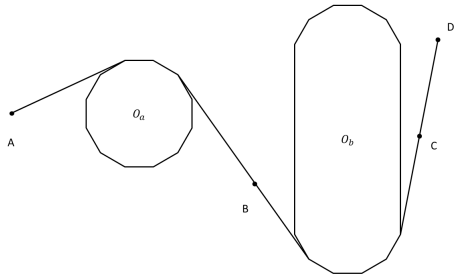

Figure 9 Trajectory created using the whole projection of O_b in the two-dimensional space.

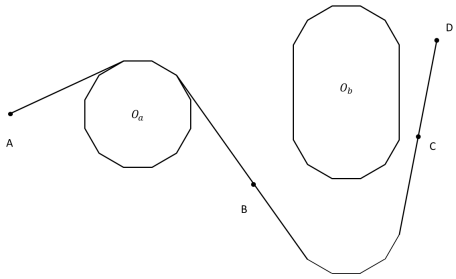

Figure 10 Projection of O_b restricted to the traveling time between points B and C.

4.2 Solution approach

4.2.1 First approach

Once obstacles are defined including their time dimension, an issue arises in computing a convex hull in order to extract two unique paths. Indeed, to do so, we need to consider obstacles in a two-dimensional space and design trajectories around them. However, it is not an easy task to determine an obstacle position when the obstacle is moving. We can for example use a pessimistic approach, which consist in avoiding an obstacle on all the space that he would cover while we go from point a point A to a point D designing the trajectory (see Fig. 9 and Fig. 10). In the example in Fig. 10, O_a is not moving and O_b has the same shape as O_a but is moving from the bottom to the top. However, when we are around the obstacle O_b (in other words, when the aircraft flying on the trajectory is on the section between the points B and C), the obstacle has moved and the previous planned trajectory (using the projection of the obstacle) is no more optimal. To solve this issue we dissociate the trajectory definition in the two-dimensional space (still building the trajectory around two-dimensional obstacles) from the evaluation through the objective function, that takes into accounts the other dimensions. The problem becomes how to transcript our three-dimensional elements into two-dimensional elements in an efficient way. A first way to address this problem is transforming a moving object into a large number of static objects existing only for a short period of time all over the trajectory of the moving object. We let the genetic algorithm decide which one of these static objects is the best one to consider. However, doing this increases dramatically the computation time to solve our problem. To figure out this problem, let Ω be a set of obstacles composed of ten static obstacles. Let O_m be a moving obstacle and $\Omega_m = \Omega \cup \{O_m\}$. Obstacle O_m appears at $t_o = 1.0$ seconds and disappears at $t_f = 5.0$ seconds. In Table 1, we report the dimension of the state space with the associated time step. To compute these values, we assume that each point of the state space is composed of n variables, where n is the number of obstacles. Each of these variables can take three values, depending on the choise of ignoring the obstacle and go through it, bypassing it in a concave way or bypassing it in a convex way. We obtain 3^n possibilities for each point of the state space. The number n when there are moving obstacles depends on the time step. Since in our example O_m exists during a 4 seconds period, if the time step is 0.5 we will need 8 obstacles to represent O_m. Let N be the number of possibilities for a chromosome, we have :

$$N = 3^{n+\frac{t_f - t_o}{\Delta t}} \tag{19}$$

where, $n = card(\Omega)$ and Δt is the time step.

4.2.2 Cluster approach

As an alternative to the above approach, we propose an approach based on clusters of obstacles. We define a cluster as a set of n obstacles, where one selects a number $m \leq n$ of obtacles to be bypassed at a given time, ignoring the others. Note that a cluster contains possible positions for the same obstacle at different times. Furthermore, an aircraft following the designed trjectory will be close to the obstacle between two times t_a and t_b, so the position of the obstacle at these two times can be used to build the trajectory. The values of times t_a and t_b are computed using the genetic algorithm presented above.

Remark that for static obstacles a cluster is defined as a set composed of only one obstacle with $m =$

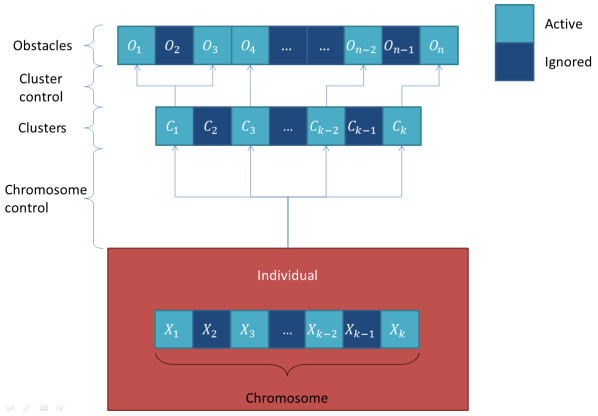

Figure 11 Management of obstacles using clusters. The dimension of the chromosome is strongly reduced with respect to the first proposed approach, while the performance of the algorithm is near the same.

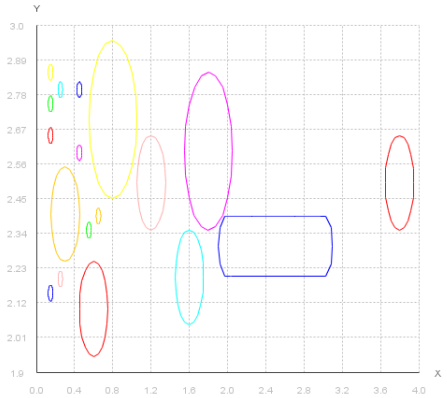

Figure 12 Obstacles in Ω. The blue obstacle has been drawn for all the duration on its appearance.

Table 2 Comparison of the cluster and the naive approach in terms of size of the state space

Time step	cluster method	naive approach
0.5	1.134×10^7	3.874×10^8
0.2	7.086×10^7	2.058×10^{14}
0.1	2.834×10^8	7.178×10^{23}

1, which corresponds to the simple two-dimensional case.

The number of possible combinations N corresponding to a system of n clusters is given by:

$$N = 3^n \prod_{i=1}^{n} r_i^{m_i} \tag{20}$$

where,
n is the number of clusters.
r_i is the number of obstacles in the ith cluster.
m_i is the number of obstacles that can be used at the same time in the ith cluster.
In Table 2, the dimensions of the state space in the above naive approach and in the cluster approach are compared, showing the drastic size reduction induced by the latter.

4.2.3 *Genetic algorithm: new chromosome definition*

Using the clusters-based approach, we need to change the meaning of the variables in a chromosome, to be used in the genetic algorithm. Indeed, in this case we do not control exactly which obstacle we are going to avoid, but we control which cluster we are going to avoid. Let Ω be a set of n obstacles that form k clusters. Let C be a cluster and G_c the set of obstacles included into C. Let m be the number of obstacles in C to be possibly avoided. When C is activated during the computation (at the initialization phase or after a mutation), the genetic algorithm choses randomly m obstacles within G_c and use them to construct the trajectory. The structure of the new chromosome is given in Fig. 11.

5 SOME RESULTS WITH MOVING OBSTACLES

Let consider an instance of the problem of designing a trajectory avoiding static and moving obstacles, like the one depicted in Fig. 12. In this example there are 18 obstacles, that are all static but one, represented in blue color, that models a moving storm. The storm appears on the left side and moves to the right. During its movement, it describes a shape that it covers at different times. Suppose that we want to design a trajectory from the origin point $(0.0, 2.5)$ to the destination point $(4.2, 2.5)$. To do so, we apply the above genetic algorithm with the parameters in Table 3. The computing time to obtain the solution is about 20 seconds on a Pentium 3GHz. The trajectory designed by the algorithm is shown in Fig. 13. This trajectory is based on intermediate states of the storm to avoid it. Displaying an animated aircraft moving along the designed trajectory we indeed see that it avoids the storm almost perfectly.

Table 3 Genetic algorithm: parameters

Population size	500
Number of generations	500
Probability of mutation ψ_1	0.20
Probability of mutation ψ_2	0.15
Probability of mutation ψ_3	0.65

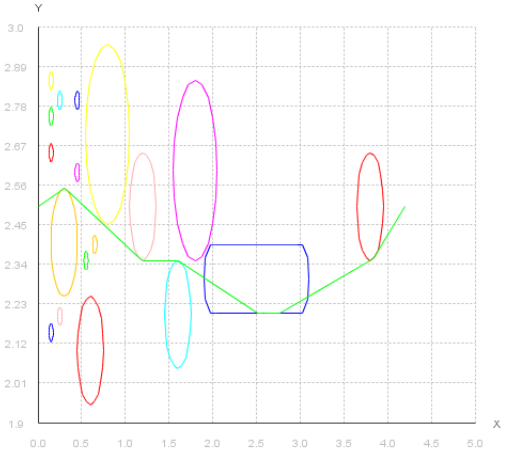

Figure 13 Designed trajectory. Remind that the blue obstacle has been drawn for its whole time extesnion. The aricraft does not intesect it at any time.

6 CONCLUSION

We presented a trajectory planner which is able to perform obstacle avoidance in an efficient way. We first adapted the Jarvis March algorithm in order to generate convex or concave obstacle avoidance, and have used it to develop a trajectory generation in a two-dimensional space that avoids obstacles while optimizing the trajectory length. The problem is modeled as a combinatorial optimization problem, whose size may be large, especially in the case of a large number of obstacles and in that of moving obstacles. Thus, we propose to address the problem with an artificial evolution approach. Specifically, we propose a genetic algorithm, where mutation operators have been developed with different granularities and applied with different probabilities. An extension based on clusters of obstacles has also been developed in order to improve the performance of the algorithm in the case of moving obstacles.

Future work will address the extension of the proposed approach for the design of trajectory shapes in a three-dimensional space, to address more closely the design of departure and arrival 3D-routes in Terminal Manoeuvering Areas.

ACKNOWLEDGMENTS

This work has been partially supported by French National Research Agency (ANR) through grant ANR 12-JS02-009-01 ATOMIC.

References

[1] H. Abdi and L.J. Williams. Principal component analysis. *Interdisciplinary Reviews: Computational Statistics*, 2,:433–459, 2010.

[2] Birkhoff and C de Boor. Piecewise polynomial interpolation and approximation. In *Proceeding of the General Motors Symposium of 1964*. General Motors, 1964.

[3] C de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[4] Daniel Delahaye and Stephane Puechmorel. *Modeling and Optimization of Air Traffic*. Wiley, 2011.

[5] Alison Eele and Richard Arthur. Path-planning with avoidance using nonlinear brach-and-bound optimization. *Journal of Guidance Control and Dynamics*, 32(2):384–394, 2009.

[6] Farin, Gerald, and Hansfordand Dianne. *The Essentials of CAGD*. A K Peters, Ltd, 2000.

[7] G Farin. *Curves and surfaces for computer aided geometric design. A practical guide*. Academic Press, 1993.

[8] T Hearth, M. *Scientific Computing, An Introductory Survey*. Computer graphics. McGraw-Hill, 2002.

[9] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2:18–21, 1973.

[10] Yimming Zhao and Panagiotis Tsiotras. A quadratic programming approach to path smoothing. *American Control Conference (ACC)*, 2011.