Strategic planning in ATM with a stochastic anytime approach

J. A. Cobano
Robotics, Vision and Control Group
Engineering School
University of Seville (Spain)
Email: jacobano@cartuja.us.es

D. Alejo
Robotics, Vision and Control Group
Engineering School
University of Seville (Spain)
Email: dalejo@cartuja.us.es

G. Heredia
and A. Ollero
Robotics, Vision and Control Group
Engineering School
University of Seville (Spain)
Email: guiller@cartuja.us.es
aollero@cartuja.us.es

***Abstract*—This paper presents a strategic trajectory de-confliction algorithm that can be applied during the pre-departure and the flight execution phases. The proposed method detects conflicts using an algorithm based on axis-aligned minimum bounding box and resolves them cooperatively using a collision-free trajectory planning algorithm based on a stochastic optimization technique named Particle Swarm Optimization (PSO). PSO modifies the trajectories of the aircraft involved with an overall minimum cost. Determining optimal trajectories with short time intervals in the flight phase is not feasible, hence an anytime approach using PSO is applied. This approach yields trajectories whose quality increases with increase in available computation time. Thus, the method could be applied to Medium-Term Conflict Detection and Resolution (MTCDR) and even Short-Term Conflict Detection and Resolution (STCDR) depending on the look-ahead times. The method has been validated with simulations in scenarios with multiple aircraft in a common airspace.**

I. Introduction

The implementation of an efficient, safe, fair and reliable Air Traffic Management (ATM) system is one of the fundamental challenges of SESAR for the future because of the increasing traffic volume in the next years [1]. The automation technologies will play an important role in this future ATM to satisfy high air traffic demand and higher levels of automation in ATM should be explored. These automation technologies should be supported by methods and algorithms to implement trajectory based operations by increasing airspace capacity and efficiency [2]. Currently this management relies on a centralized system (ATC) and the workload of the air traffic controllers will increase and the system could respond slower in the future. Thus, new decision support systems should be proposed to allow the controller to timely react to dangerous situations by facilitating collision-free trajectories. Therefore, trajectory de-confliction algorithms are required to ensure the safety of the system.

General concepts and methods on path planning could be applied in order to solve this problem. A review of these methods with a comprehensive mathematical discussion is presented in [3]. Among these methods could be highlighted: potential fields [4], graph search like A* and D* [5] and Rapidly-exploring Random Trees (RRT) [6]. Other kind of methods have been proposed such as evolutionary techniques [7] [8] [9], particle swarm optimization technique [10] and multi-objective evolutionary algorithms [11].

The problem of trajectory planning is NP-hard [12] [13]. Some differential constraints given by the model of the aircraft should be considered to make the problem tractable. Sampling-based techniques, as opposed to combinatorial planning, are usually preferred in these NP-hard problems. These planning schemes match particularly well when the solution space is hard to model or unknown a priori because of its dynamic nature. The application of evolutionary techniques or particle swarm optimization are an efficient and effective alternative for this problem.

The Conflict Detection and Resolution problem has also been studied extensively and different types of techniques have been proposed [14]. These techniques are characterized depending on the following factors: dimensions of the state information, technique for dynamic state propagation, conflict detection threshold, conflict resolution technique, maneuvering dimensions, and management of multiple UAV conflicts.

One method to resolve conflicts is based on the speed assignment. In [15] the speed profile for all the aerial vehicles involved in a collision is computed in a centralized way to solve the conflict. The method presented in [16] is based on mixed-integer linear program (MILP) and it resolves the conflict by changing speed to a large number of aerial vehicles subject to velocity change constraints. However, some conflicts cannot be solved. Other methods resolve pairwise conflicts [17] but do not consider more aircraft. [18] and [19] present a method based on mixedinteger linear program (MILP) to avoid collision. In [20] a method for multiple-aircraft conflict avoidance is proposed. It is assumed that aircraft cruise at constant altitude with varying velocities and that conflicts are resolved in the horizontal plane using heading change, velocity change, or a combination thereof. Methods based on Ant Colony Optimization (ACO) algorithm have also been proposed. In [21], the application of a game theory approach to airborne conflict resolution is presented. However, these techniques are not well suited for applications that require a high level of scalability.

In this paper, a cooperative trajectory de-confliction algorithm based on PSO has been implemented. The choice of

PSO is based on its low computational overheads and faster solution convergence compared to genetic algorithms and other evolutionary algorithms [22]. The detection algorithm is based on axis-aligned minimum bounding box. Each aircraft changes its trajectory maintaining its velocity to solve the detected conflicts collaborating with the rest of aircraft. The algorithm presents, as advantages, its low execution time and its scalability. It can be applied in the pre-departure and flight execution phase. In the later, an anytime approach is proposed to resolve the conflicts. This approach yields trajectories whose quality increases with increase in available computation time. Therefore, the algorithm can be used with low look-ahead times although the quality of the solution will not be optimal.

The paper is organized into five sections. The formulation of the problem is described in Section II. The trajectory de-confliction algorithm is explained in Section III. Section IV presents the simulations performed. Finally the conclusions are detailed in Section V.

II. Problem Formulation

The problem considered in this paper is the collision-free trajectory planning of airspace users in a dense ATM scenario. A trajectory is defined by a sequence of waypoints. The aircraft share a common airspace and the separation between aircraft should be greater than a given safety distance. It is also assumed that velocity changes are not allowed. The solution only considers the addition of intermediate waypoints. After a collision is detected, the problem is solved when a collision-free trajectory for each aircraft is computed. All aircraft cooperate to solve the problem changing their initial trajectory. The information that the system needs in order to solve the problem is the following:

- Initial trajectory of each aircraft
- Parameters of the model of each aircraft
- Initial location and goal location of each aircraft.
- Look-ahead time to know the available computation time

The objective is to find collision-free trajectories that minimize the probability of having a collision while minimizing the changes of waypoints for each aircraft. The initial and solution trajectory should have the same initial and goal locations.

III. Proposed Trajectory De-Confliction Algorithm

This section describes the blocks of the proposed algorithms to resolve the conflict. First, an algorithm based on axis-aligned minimum bounding box is implemented to detect collision. Then, a resolution algorithm based on PSO is implemented to compute collision-free trajectories.

A. Axis-aligned minimum bounding box for detection

The detection algorithm is based on axis-aligned minimum bounding box. This technique presents as advantages the low time of execution and the need of few parameters to describe the system. On the other hand, it presents two disadvantages: it is not very accurate and it depends on the coordinate axes.

Each aircraft is represented with two boxes, horizontal and vertical box, joined in order to detect the conflicts (see Figure 1). Each box is defined by the intersection of three intervals, one by axis. The measurement of the horizontal box is related to the minimum horizontal separation between aircraft and the vertical box is related to the vertical separation. Thus, the minimum separation, S, between two aircraft is defined by the dimension of both joined boxes. A collision is detected when there is an overlapping between the intervals that define each box. Thus, the 3D problem is reduced to three problems of overlapping, one in each coordinate. Let us consider the intervals in one coordinate $A = [A_i, A_e]$ and $B = [B_i, B_e]$. The condition of overlapping for this coordinate is given by:

$$(A_e > B_i) \ \Lambda \ (A_i < B_e) \tag{1}$$

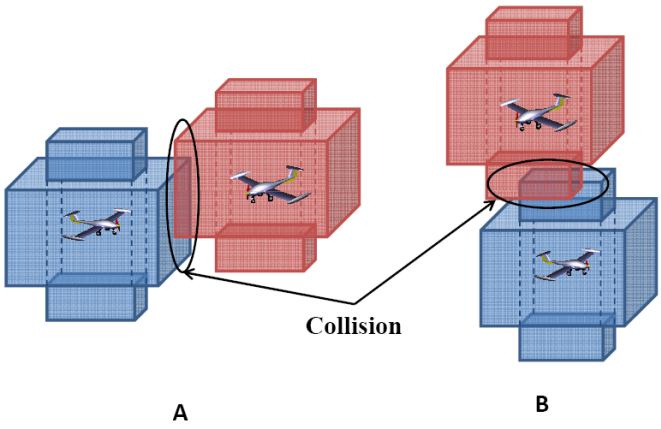

Figure 1. Detection algorithm based on axis-aligned minimum bounding box: A and B overlap (collision).

B. Resolution algorithm based on PSO

The trajectory de-confliction algorithm is based on the PSO method, which is a heuristic global optimization algorithm. This algorithm was first proposed in [23]. It is developed from swarm intelligence and is based on the research of bird and fish flock movement behavior. It works by maintaining a swarm of particles that move around in the search-space influenced by both the improvements discovered by the other particles (social behavior) and the improvements made by the particle so far (greedy behaviour). Its main advantages are its simplicity, easy implementation and the existence of few parameters to tune when comparing with other evolutionary algorithms.

In this paper, we will consider constant speed flights where the initial position is known and the final waypoint of each vehicle must be the same as the one in the original plan. Therefore, this algorithm will generate one intermediate waypoint for each vehicle so the potential collision is avoided. So, the goal of this algorithm is to obtain collision-free trajectories by adding one intermediate waypoint in the trajectory of each aircraft while minimizing the following cost function:

$$J = \sum_{i=1}^{N} L_i + \omega_c \quad (2)$$

where where L_i is the total length of trajectory ith, N is the total number of aircraft in the system and ω_c is the collision penalty that will be added if the new trajectories still lead to collisions in the system. This function can be easily modified in order to take into account energy analysis and other operational costs.

The algorithm implemented is based on in [24]. Let S be the number of particles in the swarm, each having a position $x_i \in \Re^n$ in the search-space and a velocity $v_i \in \Re^n$. Let p_i be the best known position of particle i and let g be the best known position of the entire swarm. In first place, the swarm is initialized by assigning random initial locations and velocities. A uniform distribution in the search space has been chosen for this step.

Then the exploration loop is executed. In each iteration, both the position and the velocity of each individual is updated by applying the formula indicated in steps 10 and 11 (see Algorithm 1). The most important parameters in this formula are the social weight (ϕ_g) and the local weight (ϕ_p). r_g and r_p are vectors where each component is generated at random with a $U(0,1)$ distribution. x_i and v_i are the position and velocity of individual i, p_i is the best position of particle i so far and g is the best position found so far. Local and global best positions are also updated if necessary (steps 13-15).

The exploration loop can be ended by using many differents termination criteria. In this paper, the algorithm concludes when the loop is repeated a determinate number of times. However in real-time approaches other criteria can behave better. Among these criteria a timeout condition and a convergence condition (most of the individuals lay in to a tight region of the search space) are the common approaches.

Last, the two main parameters (ϕ_g and ϕ_p) have been tuned by performing several tests with the same conditions and only changing one parameter at a time. These parameters are usually selected in the interval $[0,1]$. In our case, the best values found were $\phi_g = 0.9$ and $\phi_p = 0.1$.

Algorithm 1 Basic PSO algorithm

1. **for** Each particle **do**
2. Initialize particle position x_i with the desired probability function
3. Initialize particle best position $p_i \leftarrow x_i$
4. If $f(p_i) < f(g)$ update the swarm best position $g \leftarrow x_i$
5. Initialize the velocity vector v_i. An uniform distribution is usually used.
6. **end for**
7. **repeat**
8. **for** Each particle **do**
9. Pick random numbers r_g r_p with $U(0,1)$
10. Update the particle's velocity: $v_i \leftarrow \omega v_i + \phi_p r_p(p_i - x_i) + \phi_g r_g(g - x_i)$
11. Update the particle's position: $x_i \leftarrow x_i + v_i$
12. **if** $f(x_i) < f(p_i)$ **then**
13. Update the particle's best known position
14. **if** $f(x_i) < f(g)$ **then**
15. Update the swarm's best known position $g \leftarrow x_i$
16. **end if**
17. **end if**
18. **end for**
19. **until** A termination criterion is met

IV. Simulations

A comprehensive set of simulations have been carried out to validate the proposed method. Also, a random generation proccess of the test considered has been performed to evaluate the proposed method.

A. Test set design

The definition of a metric play on important role to evaluate the results. In cases of difficult path or motion planning problems for one only mobile object, there are some de facto benchmark standards in the academic context, like the bug trap or the alpha test [25]. However, this is not the case when dealing with multiple object planning.

In this paper, a test set has been developed in a given scenario to validate the proposed method and to provide a way to measure the properties of the method regarding time of execution, optimization and level of scalability with number of vehicles. Furthermore, the test set and the design methodology can be useful for comparison with other methods developed.

The scenario has a base of 50×50 dimensional units. Different problems are defined considering the same scenario, as well as the same random problem generation process. Each problem is formulated as a set of entry and exit points located in one of the lateral sides of the scenario.

The adopted strategy is regressive. Random candidate solutions are generated and the problem is defined using them when they are found.

The random generation process of the tests is performed following the Algorithm 2. For each vehicle, an entry side is randomly chosen, selecting a uniformly random number between 1 and 4 (line 4). Then, the exit side is randomly selected from the resting 3 sides (line 5). Entry and exit points are randomly selected from the corresponding side (line 6). A certain number, M, of intermediate waypoints inside of the scenario along with the entry and exit points define the flight plan.

The algorithm (see line 8) should ensure the following:

- The solution is valid, i.e. vehicles do not collide
- The initial plans generate a conflict, i.e. the vehicles initial plans lead to collision

The test set consists of 90,000 different problems grouped by the number of vehicles involved, from 2 to 10, in subsets of 10,000 tests. This classification, using the number of vehicles, is useful to study the scalability characteristics of the method.

Algorithm 2 Random test generation algorithm

```
for each test do
  while test is not valid do
    for each aircraft do
      Choose a random entry side
      Choose a random exit side from the resting 3
      Choose entry and exit points from the corresponding entry and exit sides
      Add M random intermediate waypoints
      Check for the flight plan validity
    end for
  end while
end for
```

B. Simulations results

Many simulations have been carried out from the test set. The tests have been performed in the same computer and under the same conditions. The results are shown in order to check the properties of the trajectory de-confliction algorithm.

The algorithms have been run in a PC with a 2GHz Dual Core processor and 2 GB of RAM. The operating system used was Kubuntu Linux with kernel 2.6.32. The code was written in C++ language and compiled with the gcc-4.4.1.

The minimum horizontal separation between aerial vehicles in XY plane is $S_{xy} = 0.8m$. The dimensions of the scenario are $50m \times 50m$. The number of intermediate waypoints, M, is set to 1. Two hundred tests have been performed for each subset.

First, we analyzes the relation between the time of execution and the number of iterations performed, and its dependence with the number of vehicles (see Figure 2). The slope usually depends on the number of vehicles and the relation is linear and additive, so each iteration should have the same computational cost.

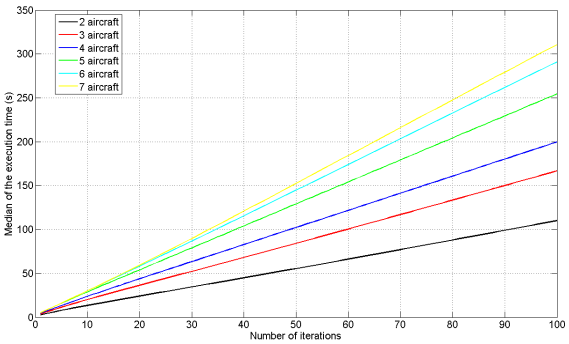

Figure 2. Time of execution vs. number of iterations depending on the number of vehicles.

Figure 3 shows the evolution of the minimum costs with the iterations. The proposed method finds a better solution as time passes, i.e. a smaller cost each iteration. The median of the minimum costs computed in the population of all the tests have been chosen as statistical indicator.

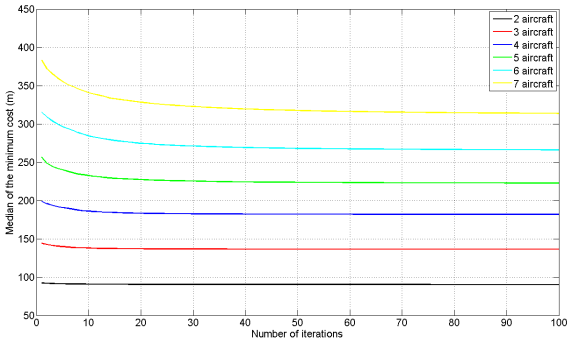

Figure 3. Median of minimum cost throughout successive iterations.

This indicator indicates how much time it would cost to achieve a solution with certain level of optimality. This relates the cost in a given iteration to the obtained minimum cost in the corresponding problem.

Table I shows the mean time of execution and the mean cost computed after 100 iterations and depending on the number of vehicles.

Table I
MEAN TIME OF EXECUTION AND MEAN COST CONSIDERING 200 SIMULATIONS FOR EACH NUMBER OF VEHICLES.

Vehicles	Mean Time of execution (s)	Mean Cost (m)
2	110.114	90.233
3	166.735	136.353
4	199.668	181.769
5	254.527	222.778
6	290.975	266.107
7	310.634	313.713

Figure 4 shows a normalization of the cost against the number of iterations. A line that marks the required number of iterations to compute for a 90% level of optimality is drawn. If the test set is executed in the same computer where the user has installed the proposed method, Figure 2 will provide an estimation of the time needed for that number of iterations, and therefore, that level of optimality.

For the cost normalization, a linear transformation, $f(x) = ax + b$, is applied to the actual cost values to set them in the range [0,1]. Therefore a and b are chosen in such a way that the maximum cost equals to 1 and the minimum cost equals to 0. Therefore,

$$a = \frac{1}{Cost_{max} - Cost_{min}} \tag{3}$$

$$b = \frac{Cost_{min}}{Cost_{min} - Cost_{max}} \tag{4}$$

Depending of the number of vehicles, a solution of great quality, 90%, is computed between 15 or 35 iterations. This means that this kind of solutions can be computed in less than one minute. This characteristic is important to apply this

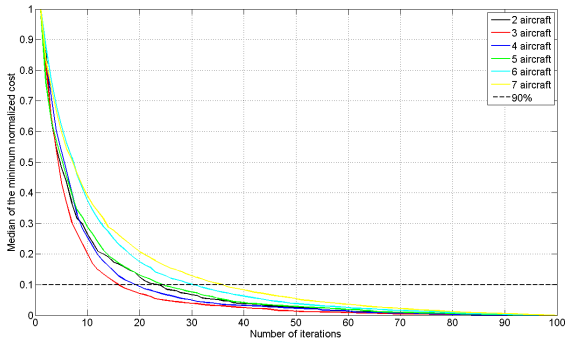

Figure 4. Normalized cost throughout successive iterations. The line marks the 90% optimality.

trajectory de-confliction algorithm in the flight execution phase even when the look-ahead time is of few minutes. Moreover, the trajectory de-confliction algorithm based on PSO presents better results than the algorithm based on genetic algorithms presented in [26].

As an example of the general operation of the proposed method, Figure 5 shows the evolution of the solution trajectories of three vehicles. The possible solution trajectory of each vehicles in a given iteration (1, 10, 30 and 60) is shown. The trajectories obtained in the same iteration are represented with the same line colour. Note that this algorithm leads to shorter trajectories as the evolution goes on.

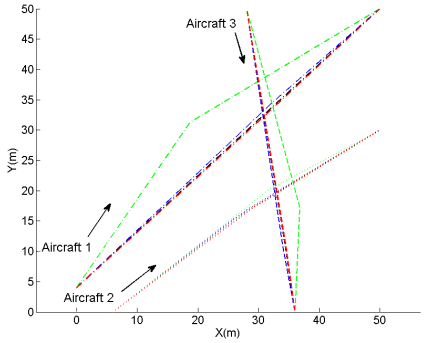

Figure 5. Evolution of solution trajectories with 3 vehicles throughout various iterations: $1th$ in green line, $10th$ in blue line, $30th$ in black line and 60th iteration in red line.

Figures 6, 7, 8, 9, 10 and 11 show the study of the anytime approach using PSO by considering two hundred simulations in each case. The results can be compared with the data shown in Figures 1 and 2, and Table I. Thus, the advantage of this approach can be observed because one good quality of solution can be obtained before one hundred iterations. The solution is reported in ten instants. The time of execution depends on the number of vehicles and the quality of solution improves as the time increases, that is, anytime approach yields trajectories whose quality increases with increase in available computation time. For example, look-ahead times of few minutes can be considered to resolve conflicts among seven aircraft.

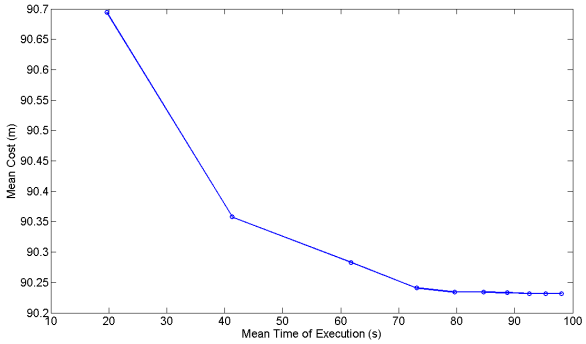

Figure 6. Anytime approach with two aircraft.

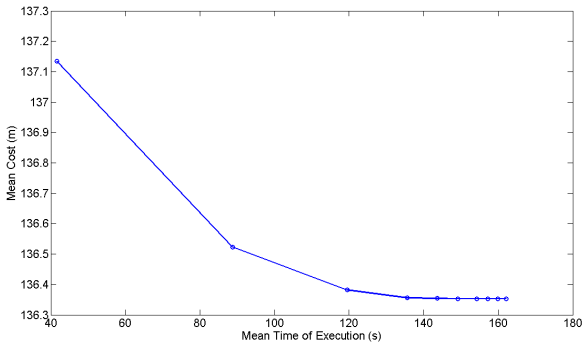

Figure 7. Anytime approach with three aircraft.

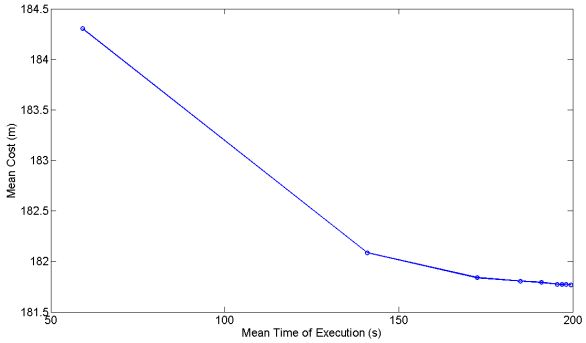

Figure 8. Anytime approach with four aircraft.

V. Conclusion

In this paper, we have presented a strategic trajectory de-confliction algorithm based on a stochastic optimization technique named Particle Swarm Optimization that can be applied during the pre-departure and the flight execution phases. The conflicts detected are resolve cooperatively and only changes of the heading of the aircraft are allowed. The algorithm has been validated with many simulations performed in different

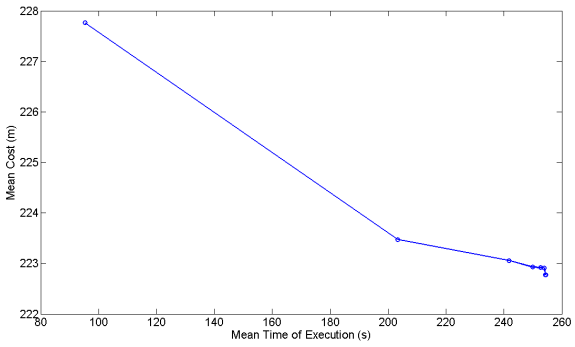

Figure 9. Anytime approach with five aircraft.

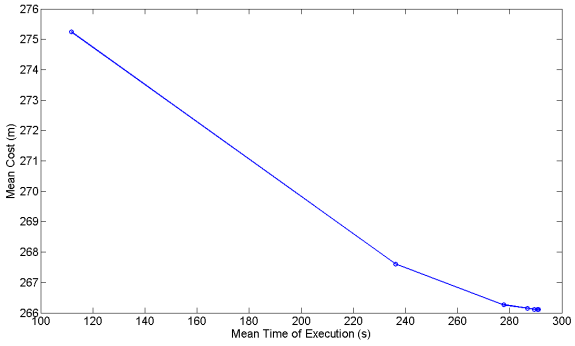

Figure 10. Anytime approach with six aircraft.

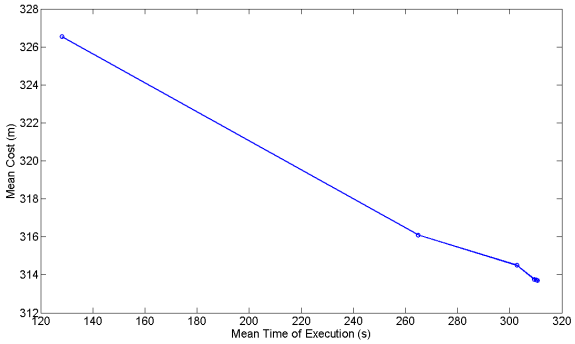

Figure 11. Anytime approach with seven aircraft.

scenarios and several studies to analyze the characteristics of the algorithm.

The main advantages of the algorithms are their low time of execution and scalability for the applications proposed. In fact, the presented algorithm improves continuously the result from a very fast first solution and results considering an anytime approach are presented. Therefore, the algorithm can be adapted to different applications that require different response times.

Future is the validation of these techniques with a larger number of aircraft (up to 10) and to performin real experiments in the CATECs testbed. Also the proposed method will take into account 4D trajectories considering times of arrival to waypoints.

ACKNOWLEDGMENT

This work was supported by the European Commission FP7 ICT Programme under the project ARCAS 287617 and the CLEAR project (DPI2011-28937-C02-01) funded by the Spanish Research and Development Program. Also the partial funding from the Junta de Andalucia project P09-TEP- 5120 is acknowledged.

REFERENCES

[1] SesarJointUndertaking, "Long-term and innovative research," in *WP-E Thematic Programme, Edition v3.0*, 15 May 2012.

[2] "Sesar consortium website:," http://www.eurocontrol.int/sesar/public/subsite_homepage/homepage.html.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.*, vol. 5, no. 1, pp. 90–98, Apr. 1986. [Online]. Available: http://dx.doi.org/10.1177/027836498600500106

[5] A. Stentz and I. C. Mellon, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments," *International Journal of Robotics and Automation*, vol. 10, pp. 89–100, 1993.

[6] S. M. Lavalle, J. J. Kuffner, and Jr., "Rapidly-Exploring Random Trees: Progress and Prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[7] J. A. Cobano, R. Conde, D. Alejo, and A. Ollero, "Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties," in *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, 2011, pp. 4429–4434.

[8] R. Vivona, D. Karr, and D.Roscoe, "Pattern-based genetic algorithm for airborne conflict resolution," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone, Colorado, August 2006.

[9] N. Durand and J. Alliot, "Optimization resolution of en route conflicts," in *First USA/Europe Air Traffic Management Research and Development Seminar (ATM1997)*, Saclay (France), 17-19 June 1997.

[10] M. Pontani and B. A. Conway, "Particle Swarm Optimization Applied to Space Trajectories," *Journal of Guidance Control and Dynamics*, vol. 33, pp. 1429–1441, 2010.

[11] A. J. Pohl and G. B. Lamont, "Multi-objective UAV mission planning using evolutionary computation," in *Winter Simulation Conference*, 2008, pp. 1268–1279. [Online]. Available: http://dx.doi.org/10.1109/WSC.2008.4736199

[12] J. F. Gilmore, "Autonomous vehicle planning analysis methodology," in *AIAAA Guidance Navigation Control Conference*, 1991, pp. 2000–4370.

[13] R. J. Szczerba, "Threat netting for real-time, intelligent route planners," in *IEEE Symp. Inf., Decis. Control*, 1999, pp. 377–382.

[14] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, pp. 179–189, 2000.

[15] J. A. Cobano, D. Alejo, A. Ollero, and A. Viguria, "Efficient conflict resolution method in air traffic management based on the speed assignment," in *Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems*, ser. ATACCS '12. Toulouse, France, France: IRIT Press, 2012, pp. 54–61. [Online]. Available: http://dl.acm.org/citation.cfm?id=2325676.2325684

[16] A. Vela, S. Solak, W. Singhose, and J.-P. Clarke, "A mixed integer program for flight-level assignment and speed control for conflict resolution," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, dec. 2009, pp. 5219 –5226.

[17] H. Erzberger, "Automated conflict resolution for air traffic control," in *Proceeding International Congress Aeronautical Sciences*, 2006, pp. 179–189.

[18] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *In Proc. ACC*, 2002, pp. 1936–1941.

[19] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 3, no. 1, pp. 3 –11, mar 2002.
[20] I. Hwang, C. J. Tomlin, I. Hwang, and C. Tomlin, "C.: Protocol-based conflict resolution for air traffic control. air traffic control quarterly 15(1," 2007.
[21] P. Masci and A. Tedeschi, "Modelling and evaluation of a game-theory approach for airborne conflict resolution in omnet++," in *Second International Conference on Dependability*, June 2009.
[22] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing, Springer*, vol. 1, pp. 235–306, 202.
[23] J. Kenendy and R. Eberhat, "Particle swarm optimization," in *IEEE Internatioal Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.
[24] M. E. H. Pedersen, "Good parameters for particle swarm optimization," in *Hvass Laboratories, Technical Report no. HL1001*, 2010.
[25] "Alpha test:," http://parasol.tamu.edu/groups/amatogroup/benchmarks/mp.
[26] D. A. A. O. J.A. Cobano, R. Conde and A. Viguria, "Conflict detection and resolution algorithm for en-route conflicts in dense non-segregated aerial traffic," in *Proceedings of the 1st International Conference on Application and Theory of Automation in Command and Control Systems, year = 2011, month = May, owner = sinosuke, address = Barcelona, Spain timestamp = 2012.01.25*.