Dynamic collision avoidance using local cooperative airplanes decisions

Augustin Degas, Arcady Rantrua, Elsy Kaddoum, Marie-Pierre Gleizes, Françoise Adreit

Dynamic collision avoidance using local cooperative airplanes decisions

Augustin Degas[1,2] · **Arcady Rantrua**[1] · **Elsy Kaddoum**[2] · **Marie-Pierre Gleizes**[2] · **Françoise Adreit**[2]

Abstract
In the near future, air traffic control (ATC) will have to cope with a radical change in the structure of air transport [1]. Apart from the increase in traffic that will push the system to its limits, the insertion of new aerial vehicles such as drones into the airspace, with different flight performances, will increase its heterogeneity. Current research aims at increasing the level of automation and partial delegation of the control to on-board systems. In this work, we investigate the collision avoidance management problem using a decentralized distributed approach. We propose an autonomous and generic multi-agent sys-tem to address this complex problem. We validate our system using state-of-the-art benchmarks. The results underline the adequacy of our local and cooperative approaches to efficiently solve the studied problem.

Keywords Trajectory optimization · Automation strategies · Conflict resolution · Self-separation · Multi-agent systems · Self-organization

1 Introduction

Contrary to a clear majority of motion planning problems, air traffic management does not rely on finding a complicated trajectory to satisfy one aircraft. Airspace is rarely cluttered by obstacles, except for the weather; thus, finding a trajectory is quite straightforward. The difficulty of motion planning for aircraft resides in finding a feasible trajectory for each aircraft (i.e. respecting the capabilities of the aircraft), collision free with the others, globally optimal (i.e., optimal for all the airplanes), and resilient to changes and uncertainties, all this in a wide configuration space [2].

✉ Augustin Degas
augustin.degas@soprasteria.com

Arcady Rantrua
arcady.rantrua@soprasteria.com

Elsy Kaddoum
elsy.kaddoum@irit.fr

Marie-Pierre Gleizes
marie-pierre.gleizes@irit.fr

Françoise Adreit
francoise.adreit@irit.fr

[1] Sopra-Steria Group, Colomiers, France

[2] IRIT, Université Toulouse 3, Toulouse, France

In today's air traffic management, airspace is divided into several zones each under the supervision of air controllers. In order to help air controllers to manage real-time traffic and avoid collisions, the traffic is regulated upstream. With the increase in traffic and the insertion of new aerial vehicles such as drones into the airspace, with different flight performances, air traffic control must evolve by increasing the level of automation and introducing partial delegation of the control to on-board systems.

In this work, we investigated this particular kind of motion planning in a clear environment, with dynamic constraints, with multiple vehicles and an objective of optimality. This kind of motion planning is mostly about following the desired trajectory while continuously avoiding collisions with other entities. We propose a generic approach that does not focus on the navigation of multiple robots to a single destination or the control of a swarm formation like in [3–5].

We propose to address the collision avoidance management problem using a decentralized distributed approach: the adaptive multi-agent system (AMAS) theory. It aims at solving problems in dynamic environments by a bottom-up design of autonomous agents, where cooperation is the driver of the self-organization process [6]. AMAS has been successfully used to solve different problems, such as anomaly detection in maritime environment [7], control and optimization of heat-engine [8], or context learning [9].

In the case of the collision avoidance problem, representing each airplane by an autonomous cooperative agent with local interactions brings a natural decentralized solution to the complexity of the global problem. The system we propose can be used for several issues:

- *Simulation* The system can be used to measure the consequences of the introduction/modification of airplanes trajectories.
- *Learning* The openness of the system allows real-time interactions with end-users. This can help for education purpose. Indeed, scenarios can be defined where the control to avoid collision is given to air traffic controllers in specific sectors and then taken back by the automated system with the modified trajectories. The proposed solution by the controllers can also be compared to the one proposed by the system.
- *On board* The system can be used by air traffic controllers or directly on board as a decision support system to avoid collisions.

The remainder of this paper is structured as follows. Section 2 briefly reviews the related work. Section 3 presents our general approach to collision avoidance. Section 4 describes its application to ATC, the experiments and the results. Finally, Sect. 5 summarizes our findings and concludes this study.

2 Related work

2.1 Problem formalization

Motion planning, planning of trajectories for a set of mobile entities, is a widely studied field, from the planning of a path of a simple robot in an unknown environment to the planning of trajectories for a set of mobile entities with constraints in a known environment. By mobile entity, we mean every possible entity that can move, from a robot arm, to cars, unmanned aerial vehicles (UAV) or airplanes. Our concerns in this vast field of motion planning lies in collision avoidance for a set of mobile entities, with dynamic constraints, and an objective of optimality.

We based our formalization on the one proposed in [2] that introduces the notion of configuration space. This notion was first introduced for path planning for a simple mobile, in order to represent the space of every possible state in which the mobile entity can be. It is easily generalized for motion planning for multiple mobile entities or a mobile entity with multiple parts, as it only had parameters related to the state vector (position, speed, direction). A configuration from the configuration space is then a vector containing a state for every $M_i \in M$ (or every part of M_i). In the remainder of this paper, the configuration space is noted C, a configuration from the configuration space is noted q. In C, some configurations q are in collision with an obstacle. We note the space of configurations with collisions with obstacles C_{Obs} and the space without collisions C_{free}; thus, $C = C_{\text{Obs}} \cup C_{\text{free}}$.

Our problem can be formalized as follows:

- A set of heterogeneous mobile entities, noted M, $M = \{M_i\}_{0<i\leq m}$
- A set of obstacles, O. An obstacle can be fixed or can move over time.

Each mobile entity is characterized by:

- A state vector, containing the position vector of M_i, its velocity vector, and the three rotation angles (Euler's angles).
- A predetermined trajectory, noted τ_p, a function of time.
- Capacities, containing its ability to accelerate, decelerate, turn and so on.
- A configuration space C_i, in which other M_j are obstacles with determined trajectories.

The problem consists in finding a trajectory τ_i for each mobile entity from a starting point, to a goal destination while respecting τ_p and avoiding collisions with mobile and stationary obstacles. In other terms, the goal is to find for each M_i a trajectory τ_i that lies in $C_{i,\text{free}}$.

2.2 State of the art

The studied problem is combinatorial: the huge number of heterogeneous airplanes, the size of the airspace and the fourth dimension (space + time) make the configuration space C tremendous. The size of the configuration space has led most research to discretize the configuration space, either for the maneuvers or the airspace, and explore it with graph search or evolutionary algorithms, or by using heuristics to guide the exploration.

Meta-heuristics using discretization of maneuvers, such as genetic algorithms [10] or ant colony algorithms [11], along with artificial intelligence algorithm as neuronal networks [12], give interesting results but they scale poorly. Indeed, [10, 11] are one of the few that handle more than 20 airplanes and still find a global optimum.

Potential fields are also expensively studied, starting from the standard method, to its extension with navigation functions [13], the usage of more complex potential fields [14], or the combination between potential fields and swarming [15]. Those methods have the particularity of providing a proof of convergence. However, those methods need to be tuned carefully to be effective. Finding generic rules to do so seems quite difficult.

Mixed-integer linear programming (MILP) solvers are studied as well, in particular for solving a minimum weight maximum clique model [16]. Results are interesting, but

they use important instantaneous heading or speed changes. Constraint programming has also shown some interesting results [17], finding solutions proved to be optimal, but also scale badly with the number of airplanes.

Geometrical approaches have also been studied. The idea is to detect collisions using velocity vectors, compute the minimal velocity vector change to avoid the collision and divide equally the minimal velocity vector change among the mobile entities [18, 19]. One algorithm in particular has been successfully applied to multi-robot [20], and some adaptations for aircraft have been made [21]. Most of them have not been tested in dense situations. The last ones however are interesting in the way they decentralize the problem among the different entities and give interesting results with dense situation as long as the constraints on maneuvers are light.

Recent studies move toward decentralization and multi-agent systems approaches [22]. Contrary to centralized approaches where a central node decides the actions of the different entities involved in the problem and insures coordination, in decentralized approaches, decision and coordination are autonomously made by the entities themselves. In such approaches, computation can be distributed on several nodes. For example, in [22], airplanes are represented by agents with the ability to modify their velocity and departure time in order to avoid collisions. Multi-agent system techniques allow a natural description of the problem and have shown their adequacy to efficiently solve complex problems addressing dynamic and scaling issues. We believe that this decentralization will be of a growing interest in the future as part of the delegation to the aircraft of responsibility for separation maintenance [1].

3 Collision avoidance using an adaptive multi-agent system approach

In this part, we start by introducing the adaptive multi-agent systems (AMAS) theory, and then, we present our general approach, called CAAMAS for collision avoidance adaptive multi-agent system.

3.1 The adaptive multi-agent systems theory

Multi-agent systems (MAS) are composed of different entities called agents. An agent is a physical or software entity, that is autonomous, evolves in an environment with perceive, decide and act abilities. The agent has a partial perception of the environment. As a result, agents do not share the same information, and do not take decisions knowing the global state of their environment. This results in a system that might make more adjustments, but is more scalable and more robust. An agent is able to communicate with other agents, has its own resources and capacities, and can offer services. An agent follows a life cycle composed of three steps, repeated until the agent achieves its local goal: perception, decision, action. During the perception, it acquires new information about its environment. It then decides the next action to perform. Then, it realizes this action. The agents' execution can be done synchronously—all agents perform in parallel a life cycle before starting a new one—, asynchronously—agents perform their life cycles without waiting others—, or iteratively (used in the experimentation conducted in this paper)—the agents execute their life cycle one after the other; the order of the agents execution is being randomly changed (or not) at each step of the system execution. From the agents local interactions, with their environment or between themselves, a self-organization is established making the solution emerge.

In some cases, an agent may be non-cooperative, which means it may bother other agents in their task, or it cannot fulfill its goal and thus cannot help the group at all. In the AMAS approach, the cooperation among agents interactions is the engine of the self-organization. The AMAS approach aims to create MAS in which agents act cooperatively between themselves in order to maintain a cooperative behavior. The AMAS theory identifies seven generic non-cooperative situations [23] among the three steps of the life cycle. In such situations, the agent has to decide cooperative actions in order to solve difficult situations (conflict, concurrence, ambiguity, etc.), based on the criticality of the agents.

3.2 The CAAMAS approach

We introduce in this section our decentralized collision avoidance adaptive multi-agent system (CAAMAS) approach, for collision avoidance management for multiple heterogeneous mobile entities with dynamic constraints. In our model, every M_i is represented by an agent following the life cycle described in Algorithm 1.

Algorithm 1 Life-Cycle of an agent M_i along its trajectory

repeat
  Perceive : Store the criticalities of mobile entities (neighbors) in its perception zone noted Zp_i;
  Decide : Compute the **criticality** of each possible action and decide cooperatively the action that minimizes the criticality of its neighbors and its own;
  Act : Perform the decided action and inform neighbors of its new criticality
until M_i has arrived

3.2.1 Perception phase

Every M_i perceives its local environment defined by its perception zone $Z\mathrm{p}_i$ of the airspace. In this zone, M_i is able to perceive other mobile entities called its neighborhood.

Different communication means can be used in order to exchange information among the mobile entities. In our model, we use messages, but other means can be easily added. In the perception phase, M_i receives messages from other mobile entities M_j ($i \neq j$). Note that every M_j perceived by M_i belongs to Zp_i ($M_j \in Zp_i$).

Those messages contain two important parts:

- The current situation of the mobile entity: its position and velocity vectors. They will be used to determine some criticalities, in particular the collision criticalities.
- The criticality of the sender, Crit_j. This criticality is computed for the action the mobile M_j is currently doing. It is used by M_i to determine its behavior regarding M_j.

During the perception phase, the agent only stores the perceived information and uses them in the decision phase.

3.2.2 Decision phase and action phase

In the decision phase, the mobile entity cooperatively decides, based on its evaluation of the current situation, which action to perform at the action phase.

(a) *Action* Given its capacities, a mobile entity M_i can realize at each step a set of finite actions in order to explore the configuration space C_i (cf. Sect. 2.1). In a fourth dimension C (space + time), those actions can be, for example, to climb, descend, stay put, turn left, turn right, accelerate or decelerate. We note Ac_i the set of n actions that M_i can do, $\text{Ac}_i = \left\{\text{Ac}_{i,k}\right\}_{0<k\leq n}$. For each possible action, the agent associates a criticality.

(b) *Criticality* For an agent, criticality represents the degree of non-satisfaction of its own goal [24]. We note Crit_i the criticality of the mobile M_i. Crit_i might be a simple real number, or a tuple of different measures. In our model, the criticality of a mobile entity M_i is a couple, $\text{Crit}_i = (\text{Crit}_{i,\text{coll}}, \text{Crit}_{i,\text{traj}})$:

- Collision criticality, noted $\text{Crit}_{i,\text{coll}}$ which represents the degree of non-satisfaction of M_i regarding the objective of avoiding collisions.
- Trajectory criticality, noted $\text{Crit}_{i,\text{traj}}$ which represents the degree of non-satisfaction of M_i regarding the objective of following its desired trajectory.

These criticalities are computed for every possible action $\text{Ac}_{i,k}$ as shown in Algorithm 2. The algorithm starts by evaluating the collision criticalities for each $\text{Ac}_{i,k}$ regarding each M_j in the perception zone, $\text{Crit}_{i,j,k,\text{coll}}$. The calculated criticalities are stored in a collision criticality list $\text{collCL}_{\text{Ac}_{i,k}}$ ordered by Crit_j associated with a $\text{Ac}_{i,k}$. Then, the trajectory criticality if M_i performs $\text{Ac}_{i,k}$, $\text{Crit}_{i,k,\text{traj}}$, is evaluated.

To sort the criticalities, the algorithm first considers the collision criticality. The trajectory criticality is considered if the collision criticality is equal or if no collision is detected.

Algorithm 2 Evaluation of the criticalities for each $Ac_{i,k} \in Ac_i$

```
Sort Crit_j of the M_j stored at the perception phase
for all Ac_{i,k} ∈ Ac_i do
    Initialize collCL_{Ac_{i,k}} associated to Ac_{i,k}
    for all perceived Crit_j do
        Evaluate Crit_{i,j,k,coll} of M_i regarding M_j if M_i does the action Ac_{i,k}
        Store the criticality in collCL_{Ac_{i,k}}
    end for
    Evaluate Crit_{i,k,traj} of M_i if it performs action Ac_{i,k}
end for
```

Evaluation of the collision criticality of an action $\text{Ac}_{i,k}$ The evaluation of the collision criticality is based on an extrapolation called nominal projection in the literature [25]. It is a simple extrapolation of position and velocity vectors. The idea is to consider that the situation of a mobile entity M_j will evolve in the same way it does currently (same speed and same direction) and thus to calculate the criticality of the possible occurrence of a collision if M_i realizes $\text{Ac}_{i,k}$.

Considering that, the collision criticality of M_i if it does $\text{Ac}_{i,k}$ regarding M_j, $\text{Crit}_{i,j,k,\text{coll}}$ is an ordered couple:

- $C_{1,k}$ representing the criticality regarding the minimal distance than can be reached between M_i and M_j if M_i realizes $\text{Ac}_{i,k}$.
- $C_{2,k}$ representing the criticality regarding the time at which the minimal distance between M_i and M_j if M_i realizes $\text{Ac}_{i,k}$ occurs.

In order to calculate this couple, the algorithm determines the minimal distance ($d_{\text{min},i,j,k}$) between M_i and M_j if M_i does the action $\text{Ac}_{i,k}$, and then the time $t_{\text{min},i,j,k}$ at which the minimal distance occurs. If a collision is detected, the interval $\left[t_{\text{startCollision}}, t_{\text{endCollision}}\right]$ of time during which the collision occurs is computed. In this case, the $t_{\text{startCollision}}$ is considered instead of the $t_{\text{min},i,j,k}$.

In the following, we note $||.||$ the Euclidean norm, and $d_{i,j,k}(t)$ the distance between M_i and M_j if M_i does the action $\text{Ac}_{i,k}$. With those notations and the previous hypothesis, we have:

$$d_{\text{min},i,j,k} = \min_{0\leq t}(d_{i,j,k}(t)) = \min_{0\leq t} ||\overrightarrow{p_{i,k}}(t) - \overrightarrow{p_j}(t)||.$$

With $\overrightarrow{p_{i,k}}(t)$ the vector position of M_i if it does the action $\mathrm{Ac}_{i,k}$, and $\vec{p_j}(t)$ the perceived position vector of M_j.

Since the speed vector is considered as constant, $d_{\min,i,j,k}$ is computed using:

$$d_{\min,i,j,k} = \min_{0 \le t} ||\left(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}}\right) - \left(\vec{p_j}(0) + t.\vec{v_j}\right)||.$$

The value at which the derivative of $||\left(\overrightarrow{p_{i,k}}(0) + t.\overrightarrow{v_{i,k}}\right) - \left(\vec{p_j}(0) + t.\vec{v_j}\right)||$ is null is then $t_{\min,i,j,k}$:

$$t_{\min,i,j,k} = \frac{-\left(\overrightarrow{p_{i,k}}(0) - \vec{p_j}(0)\right).\left(\overrightarrow{v_{i,k}} - \vec{v_j}\right)}{||\overrightarrow{v_{i,k}} - \vec{v_j}||}.$$

Note that:

$d_{\min,i,j,k} = d_{i,j,k}(t_{\min,i,j,k})$.

Based on both values, $C_{1,k}$ and $C_{2,k}$ are then calculated as follows:

$$C_{1,k} = \begin{cases} 100 - \frac{100}{2d_{\mathrm{coll}}} d_{\min,i,j,k} & \text{if } d_{\min,i,j,k} < 2d_{coll} \\ 0 & \text{if } d_{\min,i,j,k} \ge 2d_{coll}. \end{cases}$$

Where d_{coll} is the minimal distance at which two mobiles are allowed to be. Figure 1 illustrates the computation of $C_{1,\mathrm{stayPut}}$ for M_1 regarding two mobile entities M_2 and M_3 in its perception zone.

$$C_{2,k} = \begin{cases} 100 & \text{if } t_{\min,i,j,k} < t_{tr} \\ 100 - \frac{100}{2t_{tr}}\left(t_{\min,i,j,k} - t_{tr}\right) & \text{if } t_{\min,i,j,k} \in [t_{\mathrm{tr}}, 3t_{\mathrm{tr}}] \\ 0 & \text{if } t_{\min,i,j,k} \ge 3t_{\mathrm{tr}}, \end{cases}$$

where t_{tr} is the time required by the mobile to cross its perception zone.

The goal of those two criticality measures is to compute the difficulty an agent has to reach its goal. Thus, they allow the mobile entity M_i to order its different neighbors M_j from the most critical to the least critical and to help it to decide cooperatively which action to perform. In our study, both measures were normalized in the interval [0, 100] (0 corresponding to a satisfied agent, 100 to a highly critical agent).

Figure 2 represents the evolution of $C_{1,k}$ regarding d_{coll} the minimal distance at which two mobiles are allowed to be. We consider that if two mobile entities are far enough one from another ($\ge 2d_{\mathrm{coll}}$), they are satisfied ($C_{1,k} = 0$). Otherwise, the more they are close to each other, the more the $C_{1,k}$ is high.

Figure 3 represents the evolution of $C_{2,k}$ regarding t_{tr} the time required by the mobile to cross its perception zone. We consider that if a mobile entity has enough time to cross its perception zone ($\ge 3t_{\mathrm{tr}}$) before the minimal distance between two mobile entities is reached, then they are

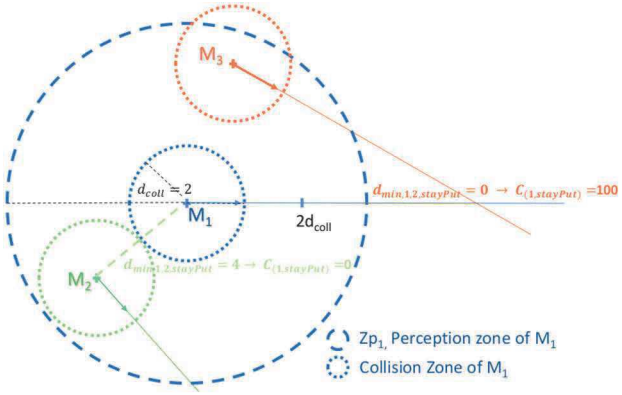

Fig. 1 Main notations used in the computation of criticalities

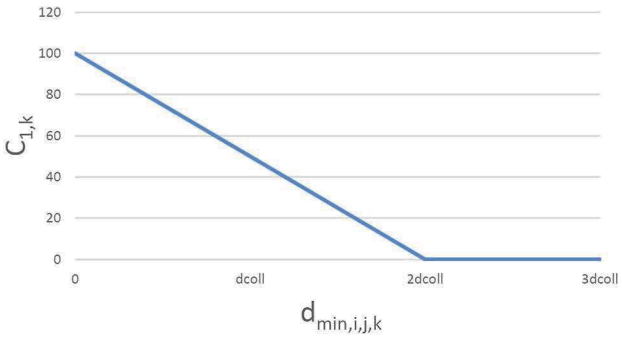

Fig. 2 Criticity $C_{1,k}$ in function of $d_{\min,i,j,k}$

satisfied ($C_{2,k} = 0$). Otherwise, more this minimal distance is reached in a near future, more the situation is urgent.

Note that, the two parameters d_{coll} and t_{tr} were decided with ATM experts to underline the nature (critical or not) of a situation.

Evaluation of the trajectory criticality of an action $\mathrm{Ac}_{i,k}$ The trajectory criticality is composed of three measures, $\mathrm{Crit}_{i,k,\mathrm{traj}} = (C_{3,k}, C_{4,k}, C_{5,k})$ where:

- $C_{3,k}$ is the distance to the predetermined trajectory τ_p at the next step (t+1) if $\mathrm{Ac}_{i,k}$ is performed at the next step (t+1)

  $$C_{3,k} = \min_{\vec{x} \in \tau_p} \left\{||\overrightarrow{p_{i,k}}(t+1) - \vec{x}||\right\}.$$

- $C_{4,k}$ is the distance to the position of the destination $\overrightarrow{p_{\mathrm{goal}}} \in \tau_p$ if $Ac_{i,k}$ is performed.

  $$C_{4,k} = ||\overrightarrow{p_{i,k}}(t = \tau) - \overrightarrow{p_{\mathrm{goal}}}||.$$

- $C_{5,k}$ is the angle α_k between the predetermined trajectory τ_p and the speed vector if $\mathrm{Ac}_{i,k}$ is performed,

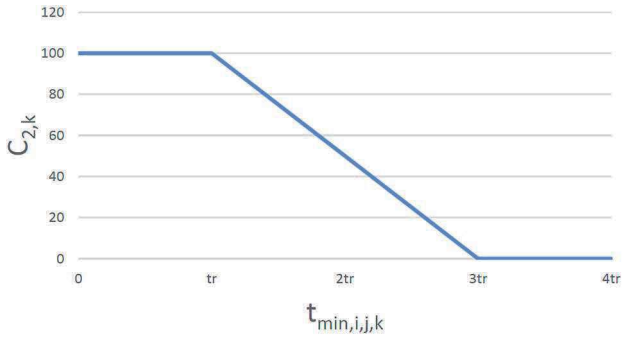

Fig. 3 Criticity $C_{2,k}$ in function of $dt_{\min,i,j,k}$

$$C_{5,k} = \begin{cases} 0 & \text{if } \alpha_k < \frac{\pi}{2} \\ \frac{100}{\frac{\pi}{2}} \times \left(\alpha_k - \frac{\pi}{2}\right) & \text{if } \alpha_k \geq \frac{\pi}{2} \end{cases}.$$

The three measures are used by the agent regarding the current situation. Two cases are to be distinguished:

- Previous trajectory modifications lead the agent to deviate from its predetermined trajectory τ_pn but still keeping the same direction. The agent compares the trajectory criticality between two possible actions using first $C_{3,k}$, then $C_{4,k}$ (for equal $C_{3,k}$) and finally $C_{5,k}$ (for equal $C_{3,k}$ and $C_{4,k}$).
- Previous trajectory modifications lead the agent to deviate from its predetermined trajectory τ_p, but with opposite direction. The agent compares the trajectory criticality between two possible actions using first $C_{5,k}$, then $C_{3,k}$ (for equal $C_{5,k}$) and finally $C_{4,k}$ (for equal $C_{5,k}$ and $C_{3,k}$).

Decide which action to perform After evaluating the criticalities of all $\text{Ac}_{i,k} \in \text{Ac}_i$, the agent cooperatively decides which action to perform in order to help the most critical agents. The decision process presented in Algorithm 3 distinguishes between the two cases: A collision is detected or not.

If a collision is detection, the idea is to consider the criticalities of every $M_j \in Zp_i$ from the highest to the lowest, and determine, using the criticality collision lists $\text{collCL}_{\text{Ac}_{i,k}}$ of each $\text{Ac}_{i,k}$ computed in Algorithm 3, which action(s) can help the most.

If several actions can be done to help most critical agents or no collision is detected, the agent decision is based on the trajectory criticality as defined above.

Algorithm 3 Decide

Create and initialize a list l_{act} with every $Ac_{i,k}$
if Collision detected **then**
  while $length(l_{act}) \neq 1$ and not all $Crit_j$ considered **do**
    Select M_j with the highest $Crit_j$
    Using $collCL_{Ac_{i,k}}$, find the set of actions $a \subset Ac_i$ that improves the criticality for M_j the most
    Remove from l_{act} every $Ac_{i,k} \notin a$
    Remove $Crit_j$ from not considered criticalities
  end while
end if
Choose the $Ac_{i,k} \in l_{act}$ that reduces the trajectory criticality $Crit_{i,k,traj}$ the most.

Note that, for a given mobile entity, the most critical agent might be an agent $M_j \in \text{Zp}_i$ or itself.

Once the action $\text{Ac}_{i,k}$ to perform is decided, the agent determines its criticality as

$$\text{Crit}_i = \left(\max\left(\text{collCL}_{\text{Ac}_{i,k}}\right), \text{Crit}_{i,k,\text{traj}}\right). \quad (1)$$

Action phase The action part for the agent is straightforward. M_i does the action decided by Algorithm 3, and sends messages containing its current situation and its criticality to the mobile entities inside its perception zone.

4 Experiments and results

To validate our model, we apply it on air traffic management (ATM): Mobile entities are airplanes with realistic characteristics.

4.1 Air traffic control

We briefly introduce here some characteristics of air traffic control (ATC) which explains the instantiation of the CAAMAS approach.

In ATC, airplanes are separated for safety by a protection zone determined using different human and material factors. The protection zone is a cylinder oriented vertically with a height h of 1000 feet (1000 ft = 304.9 m) and a radius r of 5 nautical miles (r = 5 NM = 9.26 km)

Airplanes fly from geographical points to geographical points, called waypoints. Their trajectory from one airport to another is then reduced to a list of waypoints, called flightplan. Experience shows that they do not always follow their flightplan [26] for different reasons such as controller orders or weather. In our study, we consider that their predetermined trajectory τ_p is approximated by a broken line.

Airplanes are increasingly capable of communicating data to each other such as their positions, heading or speed by means of messages. Messages can be transmitted using automatic dependant surveillance-broadcast (ADS-B).

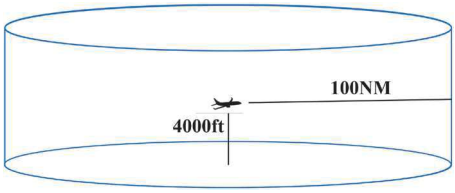

Fig. 4 Perception zone of the black airplane

4.2 Instantiating CAAMAS

(a) *Perception zone* The neighborhood of the airplane is a cylinder of radius r_{Z_p} of 100 nautical miles (100 NM = 182.5 km) and a height of h_{Z_p} of 4000 feet (4000 ft = 1219.2 m) centered on it (Fig. 4). The radius of the perception zone is the half of the horizontal range of ASD-B, in order to take into account possible congestions or loss of signals. The height allows the airplane to perceive two airways above and two airways below it, based on the most common actual structure of the air traffic management. We consider that every obstacle O_j (mobile M_j or stationary) in the cylinder is perceived by M_i. In the conducted experimentation, only mobile obstacles M_j were considered.

(b) *Collision detection* In ATC, two airplanes are considered in conflict whenever a violation of the protection zone is detected. For the sake of clarity, in the following an ATC loss of separation will be refereed as a collision, and is considered as a collision by CAAMAS. In order to take into account the fact that the vertical distance is not on the same scale as the horizontal distance, the collision criticality is calculated on the horizontal plan and the vertical plan. Thus, we have two values for $d_{\min,i,j,k}$: $d_{\min,i,j,k,h}$ and $d_{\min,i,j,k,v}$, and two values for time $t_{\min,i,j,k}$ (or an interval): $t_{\min,i,j,k,h}$ and $t_{\min,i,j,k,v}$ with every other M_j for every $\text{Ac}_{i,k}$. Then, the collision criticality $\text{Crit}_{i,j,k,\text{coll}}$ regarding M_j for action $\text{Ac}_{i,k}$ is calculated as follows:

- $C_{1,k} = \min(C_{1,k,v}, C_{1,k,h})$
- $C_{2,k} = \min(C_{2,k,v}, C_{2,k,h})$,

where $C_{1,k,v}$ and $C_{2,k,v}$ being, respectively, the equivalent of the previous $C_{1,k}$ and $C_{2,k}$ for the vertical plan, and $C_{1,k,h}$ and $C_{2,k,h}$ for the horizontal plan.

(c) *Actions* Airplanes can modify their speed (or not), by accelerating or decelerating, but also change their heading (or not), and modify their altitude as well (or not). These actions have a fixed parameter such as an acceleration rate or a turning rate. We consider that they can realize those three changes at the same time, which means that at every step of $\Delta t = 1s$ an airplane has to choose between three actions for each, that is 27 possibilities ($3 \times 3 \times 3$). This results in 27 possible future positions represented in Fig. 5. In this study, we only experiment in the horizontal plane,[1] so the number of actions available for the airplane if we authorize speed changes is only of 9 (3×3) as in Fig. 6.

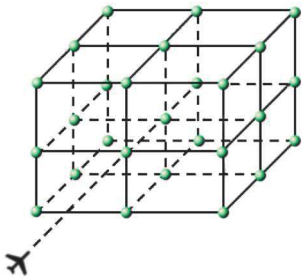

Fig. 5 The possible actions of the airplane (not to scale)

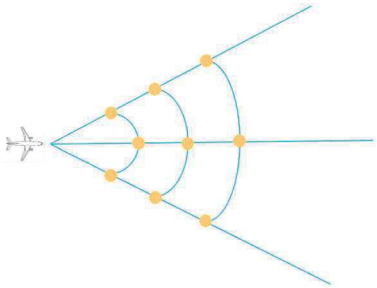

Fig. 6 The possible actions of the airplane in the horizontal plane (not to scale)

Each airplane has a preferred cruise speed depending on general airplane performances and airline preferences that we call v_{pref}. The airplanes are able to decelerate and accelerate within a speed range of $[v_{\text{pref}} - 6\%, v_{\text{pref}} + 3\%]$ which are considered as plausible values in real life [27]. It can accelerate and decelerate with speed modification of 0.33% at each step. In the experiment, we consider that airplanes have the same v_{pref} and that an airplane can modify its heading by $3° \text{ s}^{-1}$ [28, p. 18], which makes a complete 360° turn in 2 min.

4.3 Benchmark

We experiment our approach with two benchmarks, a roundabout and a random case like in [29]. We experimented

[1] This slightly simplifies the experimental setup while also providing problems that are harder to solve, as the agents are more constrained.

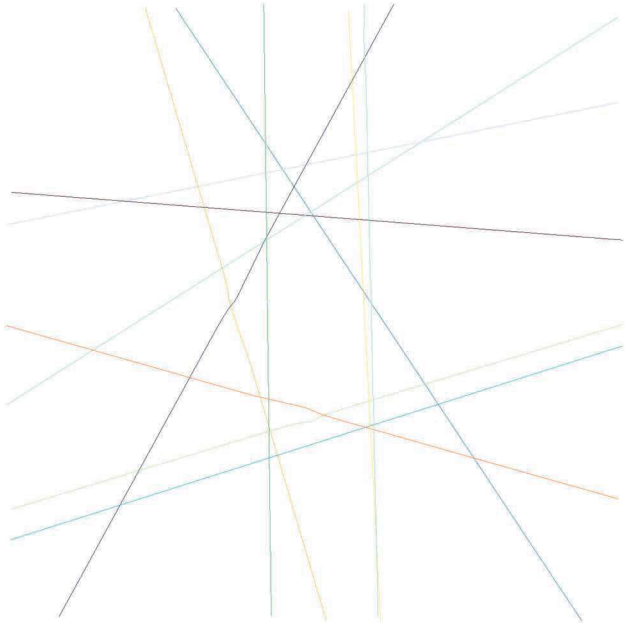

Fig. 7 Random experiment with 12 airplanes: 12 arrived and 0 collision

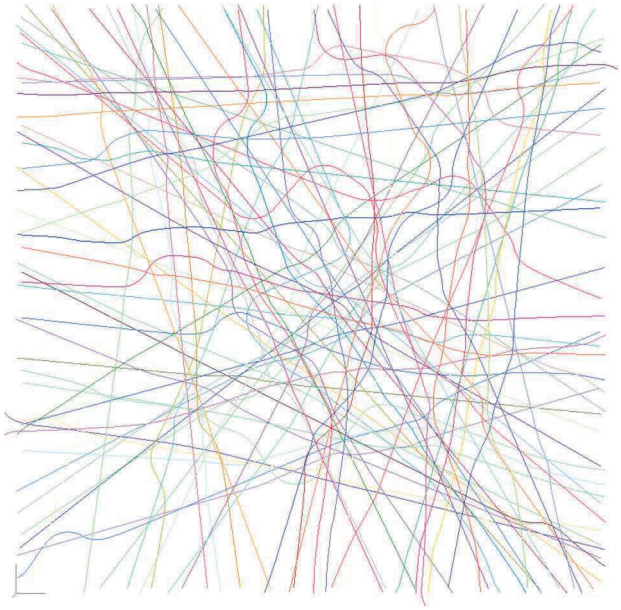

Fig. 8 Random experiment with 80 airplanes: 76 arrived and 0 collision

with several numbers of agents AgNb: for the roundabout AgNb $\in \{6, 8\}$ and for the random benchmark AgNb $\in \{12, 20, 40, 52, 60, 80, 100, 120\}$. In both experiments, airplanes cannot change their altitude.

In the roundabout benchmark, we experiment on a disk of radius $R = 125NM = 231,5$ km. At the beginning of the experiment, all the airplanes are placed at the edge of the disk and they are all converging toward the center of the disk with an angle of $\frac{2\Pi}{m}$ between them. For this experiment, only heading changes are authorized, and speed is normalized.

For the random experiment, airplanes are equally placed at each side of the square of side $l = 500$ NM. They are placed randomly and have a precise point on the opposite side of the square as destination like in Figs. 7 and 8. The arrival and start distance must be at a distance $d > 2d_{\text{coll}}$ from each other.

We evaluate our experiments regarding the number of predicted collisions, the number of remaining collisions, the computation time, and the delays caused to airplanes. Comparison with the results obtained in [29] underline the advantages of our approach.

4.4 Results

In this section, separated results for the random and the roundabout experiments are presented. The tests were performed on a computer equipped with a 2.50 GHz i7-6500 processor and 8 GB of RAM. We implemented the algorithms in Java 8. Due to the decentralized and distributed nature of multi-agent systems that introduce some indeterminism in the resulting solutions, our system has been tested 100 times per *AgNb* for both random and the roundabout-conducted experiments. The mean computed results are presented in the next sections.

4.4.1 Random experiment

For this experiment, the AgNb $\in \{12, 20, 40, 52, 60, 80, 100, 120\}$. The evaluated metrics are presented each in separated figures as box-plots for every AgNb, with maximum value, third quartile, median, first quartile and minimum value. Table 1 summarizes the comparative study.

(a) *Predicted and remaining collisions* For each randomly generated scenario, we compute the number of predicted collisions (Fig. 9), i.e., the number of collisions that would occur if no change is made on the trajectories of the airplanes. We compare this number to the remaining collisions (Fig. 10) that may be new collisions created by the system while avoiding others, or the same old collisions.

The obtained results show that in most cases more than 88.4% of the collisions have been solved (more than 99.12% for 40 airplanes). Still, detailed studies must be conducted in order to count the number of new collisions added by solving predicted collisions.

(b) *Computation time* Fig. 11 shows the time the system uses to compute the trajectory of every airplane. Note

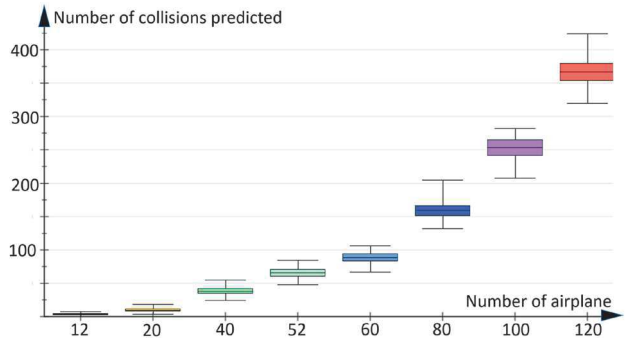

Fig. 9 The number of collisions predicted at the start of the test

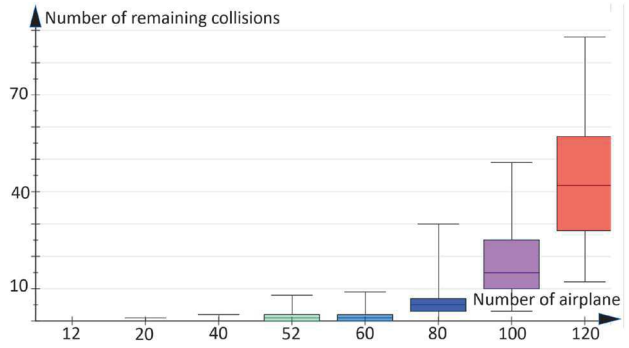

Fig. 10 The number of remaining collisions after the test

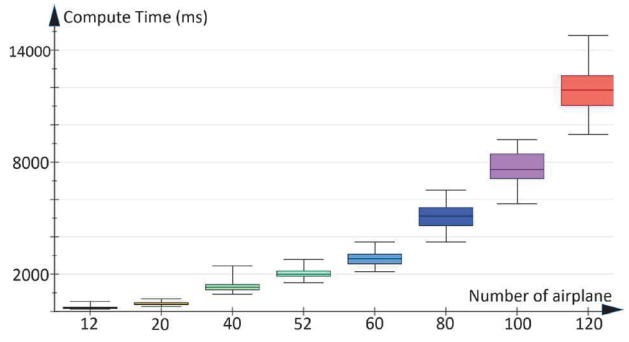

Fig. 11 Time to compute the solution in milliseconds

that, for the benchmark with 120 airplanes, the algorithm takes less than 12 s to compute the solution. In average, it takes 212 ms, 418 ms, 1305 ms, 2048 ms, 2826 ms, 5072 ms, 7660 ms and 11891 ms to compute the algorithm with, respectively, 12, 20, 40, 52, 60, 80, 100, 120 airplanes. Note that CAAMAS can be deployed on a distributed computation network which can drastically reduce the computation time.

(c) *Delays caused to airplanes* Fig. 12 shows the results concerning the delays caused to airplanes in order to avoid collisions. The results show that for each benchmark, more than 75% of the airplanes have very short delays. Still in high-density benchmarks (i.e., 120 airplanes),some airplanes can be delayed significantly. This can be due to the limited nine actions given to the airplane. Still, a detailed study will be conducted in order to understand the characteristics of such benchmarks (density, new generated collisions while solving predicted ones, etc.) and to be able to propose better trajectory for highly delayed airplanes.

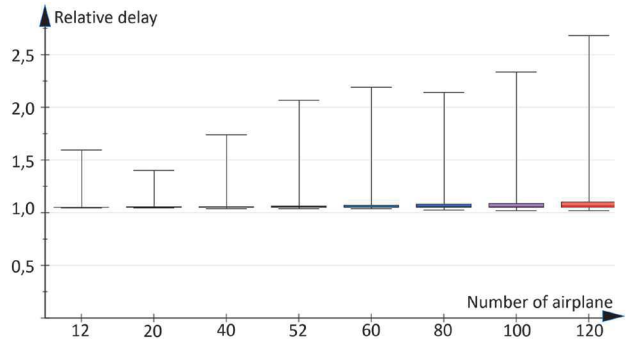

Fig. 12 Delays caused to airplanes

(d) *Comparative study* Table 1 summarizes the comparisons realized between CAAMAS and [29].

For the comparative study, the speed range has been set to $[v_{\text{pref}} - 5\%, v_{\text{pref}} + 5\%]$. An aircraft can still accelerate and decelerate with speed modifications of 0.33% at each step and modify their heading by 3° s^{-1}. We compare with the results of Table I and Table II from [29], with $s_{\text{lat}} = 0.05$ and $s_{\text{long}} = 0.05$.

For the slight scenarios (10 and 20 airplanes) presented in Table 1 CAAMAS solves all the collisions, while the comparative algorithm is not able to solve some of them. When the number of airplanes increases too much (from 100 to 120), the number of collisions increases significantly for both, but Durand et al. [29] do not give delays since the algorithm cannot bring any aircraft to its destination. The last scenario with 120 airplanes could be considered as very overloaded because the collision number increases significantly. The max delay is quite high (from 50 to 140%) compared to the Durand et. al. [29] or reality, and represent isolated aircraft. Nevertheless, the mean flight delay is not so high (10%).

4.4.2 Roundabout experiment

For this experiment, we defined a benchmark with, respectively, 6, 8 and 10 airplanes with specific trajectories to simulate the roundabout experiment.

The evaluated metrics (mean delay, standard deviation, first and third quartile) are presented in Table 2. The mean

Table 1 Comparative study (dimensionless measures)

NB acft	CAAMAS			Durand et. al. [29]		
	Rem Coll	Mean delay	Max delay	Rem Coll	Mean delay	Max delay
10	0	1.04504	1.58842	0	1.00094	1.00898
20	0	1.04745	1.56263	2	1.00431	1.03019
50	1	1.06049	1.91479	16	1.01376	1.03342
100	18	1.08122	2.05103	92	Void	Void
120	38	1.08897	2.39579	100	Void	Void

Table 2 Relative airplane delays due to collision avoidance

NB acft	Mean delay	Std. deviation	Q_1	Q_3
6	1.00735	0.01210	1.0	1.013531799729364
8	1.01310	0.02028	1.00000	1.02029
10	1.02291	0.03447	1.00338	1.03721

relative delays are acceptable. The standard deviation underlines the fact that the relative delay is not always equally distributed among airplanes. This is due to the implementation made for this experimentation in which airplanes (agents) decide iteratively; thus, the first-executed airplanes are more penalized than the others. Finally, in our approach, agents return to their trajectory once the collision is avoided before going to their destination, while in most algorithms airplanes go directly to the destination. This can increase the relative delay caused to airplanes.

Another interesting point in this experiment is the emergence of the usual pattern called roundabout expected to avoid collision in such scenario. Figures 13, 14 and 15, respectively, show the obtained trajectories for 6, 8 and 10 airplanes.

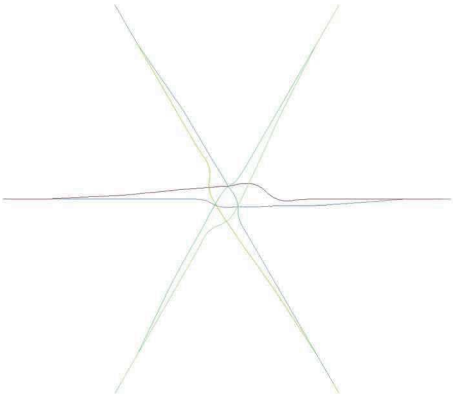

Fig. 13 Near-optimal solution for 6 airplanes

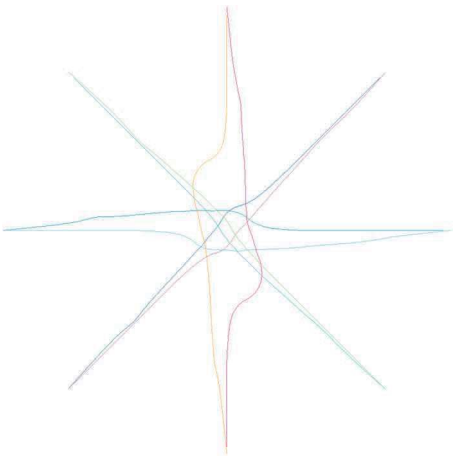

Fig. 14 A solution for 8 airplanes

5 Conclusion

The proposed collision avoidance system is a fully decentralized distributed approach based on adaptive multi-agent technology. We have shown its relevance with several criteria: efficiency of management even for dense traffic, limited amount of communication between airplanes, and small computation time.

In this system, each airplane is an agent that decides by itself of its trajectory giving its local point of view. As shown in Fig. 16, one airplane will only perceive a restricted number of airplanes (those in its perception zone described by Fig. 4) per step of life cycle, and only partial information about its neighbors (as explained in Sect. 3.2.1). This kind of decentralization brings more resilience to the system, and should allow to take into account non-cooperative obstacles in the scenarios, such as airplanes unable to avoid others or weather.

Moreover, since CAAMAS is naturally distributed and decentralized, it could eventually be implemented on board, removing the need to rely on ground equipment. As it does not rely on a computed plan, we can assert that our method is fully adaptive when facing unexpected events. For example, an airplane could become highly critical, due to a loss of an engine, and others would take this new event and this new criticality into account when deciding their trajectories.

Nevertheless, despite the encouraging obtained results concerning the remaining collision and the relative delays,

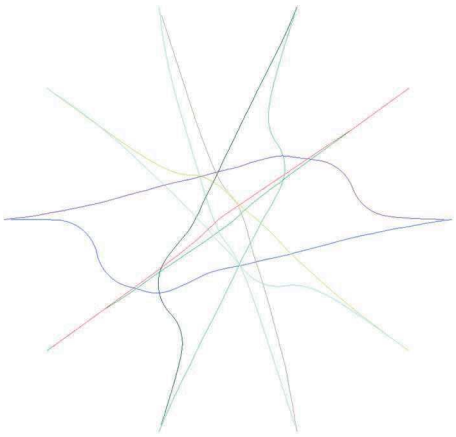

Fig. 15 A solution for 10 airplanes

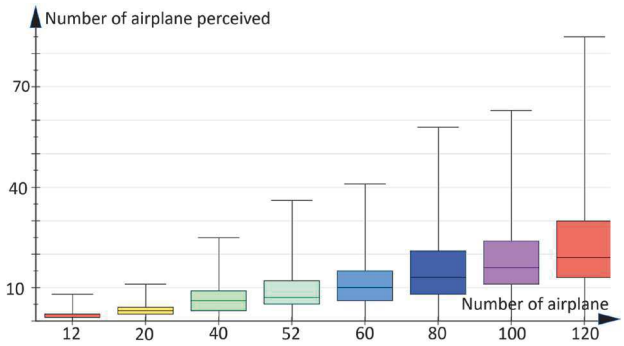

Fig. 16 Number of perceived airplanes at one step by one airplane during the different random experiments

CAAMAS is unable to solve all conflicts for overloaded scenarios. Experiments with additional possible actions for airplanes will be conducted. A further study of the impact of the density of the scenario must be done.

We also plan to test our algorithm when some communications are lost, and adding non-cooperative obstacles in the scenarios, such as airplanes or weather, to prove it resilience and to experiment the benefits of having a percentage of cooperative airplane in the traffic. Testing other means of perception, such as radar, would also be an interesting case study.

To more thoroughly evaluate the performances of CAAMAS, we intend to implement a standard global optimization method—commonly used for solving similar problems—in order to compare the difference in terms of computation time and result optimality.

Eventually, CAAMAS could be used as a support decision system for air traffic controllers because it is able to work over different scales of time and space. This would require comprehensive experiments with data obtained from real air traffic.

Acknowledgements The authors thank Sopra Steria Group and the ANRT for their support in this research work.

Funding This work was supported by the Association Nationale de la Recherche et de la Technologie (Grant no. 2016/0940).

References

1. Prandini, M., Piroddi, L., Puechmorel, S., Brázdilová, S.L.: Toward air traffic complexity assessment in new generation Air Traffic Management systems. IEEE Trans. Intell. Transp. Syst. **12**(3), 809–818 (2011). [Online]. https://hal-enac.archives-ouvertes.fr/hal-01020894
2. Latombe, J.-C.: Robot Motion Planning, vol. 124. Springer Science & Business Media, Berlin (2012)
3. Antonelli, G., Arrichiello, F., Caccavale, F., Marino, A.: Decentralized centroid and formation control for multi-robot systems. In: Robotics and Automation (ICRA), 2013 IEEE international conference on. IEEE, pp. 3511–3516 (2013)
4. Huang, C., Chen, X., Zhang, Y., Qin, S., Zeng, Y., Li, X.: Hierarchical model predictive control for multi-robot navigation. In: Brewka, G. (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16), pp. 3140–3146. AAAI Press
5. Tian, Y., Sarkar, N.: Formation control of mobile robots subject to wheel slip. In: Robotics and automation (ICRA), 2012 IEEE International Conference on. IEEE, pp. 4553–4558 (2013)
6. Serugendo, G.D.M., Gleizes, M.-P., Karageorgos, A.: Self-Organising Software: From Natural to Artificial Adaptation. Springer Science & Business Media, Berlin (2011)
7. Brax, N.: Self-adaptive multi-agent systems for aided decision-making: an application to maritime surveillance, Ph.D. dissertation, Université de Toulouse, Université Toulouse III-Paul Sabatier (2013)
8. Boes, J., Migeon, F., Gatto, F.: Self-organizing agents for an adaptive control of heat engines. ICINCO **1**, 243–250 (2013)
9. Nigon, J., Gleizes, M.-P., Migeon, F.: Self-adaptive model generation for ambient systems. Procedia Comput. Sci. **83**, 675–679 (2016)
10. Delahaye, D., Peyronne, C., Mongeau, M., Puechmorel, S.: Aircraft conflict resolution by genetic algorithm and b-spline approximation. In: EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS, pp. 71 (2010)
11. 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 development seminar (2009)
12. Christodoulou, M.A., Kontogeorgou, C.: Collision avoidance in commercial aircraft free flight via neural networks and non-linear programming. Int. J. Neural Syst. **18**(05), 371–387 (2008)
13. Roussos, G., Kyriakopoulos, K.J.: Completely decentralised navigation functions for agents with finite sensing regions with application in aircraft conflict resolution. In: Decision and control and european control conference (CDC-ECC), 2011 50th IEEE Conference on. IEEE, pp. 7470–7475 (2011)
14. Guys, L., Puechmorel, S., Lapasset, L.: Automatic conflict solving using biharmonic navigation functions. Procedia Soc. Behav. Sci. **54**, 1378–1387 (2012)
15. Maas, J., Sunil, E., Ellerbroek, J., Hoekstra, J.: The effect of swarming on a voltage potential-based conflict resolution algorithm. In: submitted to the 7th international conference on research in air transportation (2016)

16. Lehouillier, T., Nasri, M.I., Soumis, F., Desaulniers, G., Omer, J.: Solving the air conflict resolution problem under uncertainty using an iterative biobjective mixed integer programming approach. Transp. Sci. **51**, 1242–1258 (2017)
17. Allignol, C., Barnier, N., Durand, N., Alliot, J.-M.: A new framework for solving en-routes conflicts. In: ATM 2013, 10th USA/Europe Air Traffic Management Research and Development Seminar, Chicago, United States, pp 1–9. [Online]. https://hal-enac.archives-ouvertes.fr/hal-00828736 (2013)
18. Machado, P., Bousson, K.: Automatic collision avoidance system based on geometric approach applied to multiple aircraft. In: 6th international conference on research in air transportation (ICRAT 2014) (2014)
19. Lin, C.E., Lee, C.-J.: Conflict detection and resolution model for low altitude flights. In: Methods and models in automation and robotics (MMAR), 2015 20th International Conference on. IEEE, pp. 406–411 (2015)
20. Van Den Berg, J., Guy, S., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. Robot. Res. **2011**, 3–19 (2009)
21. Allignol, C., Barnier, N., Durand, N., Blond, E.: Detect & avoid, uav integration in the lower airspace traffic. In: ICRAT 2016, 7th international conference on research in air transportation (2016)
22. Breil, R., Delahaye, D., Lapasset, L., Féron, É.: Multi-agent systems for air traffic conflicts resolution by local speed regulation. In: 7th international conference on research in air transportation (ICRAT 2016) (2016)
23. Capera, D., Georgé, J.-P., Gleizes, M.-P., Glize, P.: The amas theory for complex problem solving based on self-organizing cooperative agents. In: Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on. IEEE, pp. 383–388 (2003)
24. Bonjean, N., Mefteh, W., Gleizes, M.P., Maurel, C., Migeon, F.: Adelfe 2.0. In: Handbook on agent-oriented design processes. Springer, pp. 19–63 (2014)
25. Kuchar, J.K., Yang, L.C.: A review of conflict detection and resolution modeling methods. IEEE Trans. Intell. Transp. Syst. **1**(4), 179–189 (2000)
26. Rantrua, A., Maesen, E., Chabrier, S., Gleizes, M.-P.: Learning aircraft behavior from real air traffic. J. Air Traffic Control **57**(4), 10–14 (2015)
27. Averty, P., Johansson, B., Wise, J., Capsie, C.: Could erasmus speed adjustments be identifiable by air traffic controllers. In: 7th USA/Europe air traffic management research and development seminar (ATM2007), vol. 22 (2007)
28. Authority, F.A.: Aeronautical information manual: Official guide to basic flight information and ATV procedures (2006)
29. Durand, N., Barnier, N.: Does atm need centralized coordination? Autonomous conflict resolution analysis in a constrained speed environment. Air Traffic Control Q. **23**(4), 325–346 (2015)