Multi-agent Systems for Air Traffic Conflicts Resolution by Local Speed Regulation and Departure Delay

Romaric Breil, Daniel Delahaye, Laurent Lapasset, Eric Féron

Multi-agent Systems for Air Traffic Conflicts Resolution by Local Speed Regulation and Departure Delay

Romaric Breil, Daniel Delahaye
Laboratory in Applied Mathematics, Computer Science and Automatics for Air Transport
National Civil Aviation University
Toulouse, France
E-mail: breil@recherche.enac.fr

Laurent Lapasset
Capgemini Technology Services
Toulouse, France

Éric Féron
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332 USA

***Abstract*—Air Traffic Flow Management (ATFM) aims at structuring traffic in order to reduce congestion in airspace. Congestion being linked to aircraft located at the same position at the same time, ATFM organizes traffic in the spatial dimension (e.g. route network) and/or in the time dimension (sequencing and merging in TMA, Miles-in-Trail for en-route airspace).**

The objective of this paper is to develop a methodology that allows the traffic to self-organize in the time dimension when demand is high. This structure disappears when the demand diminishes.

In order to reach this goal, a multi-agent system has been developed. In this system, aircraft agents regulate speed and delay departure time in order to reduce the number of conflicts, thus decreases overall traffic complexity, which becomes easier to manage by air traffic controllers. This algorithm was applied on realistic examples.

***Index Terms*—Air traffic management, multi-agent system, conflict resolution, speed regulation, departure delay**

I. Introduction

Air traffic volume has been constantly increasing during the past decades, and ICAO [1] predicts that the annual number of flights will double in 2030 in comparison to 2013. Air traffic controllers are in charge of ensuring traffic safety and fluidity by temporarily diverting flights from their original trajectory when necessary. In doing so, a minimum separation distance is maintained between all aircraft. This task is known as conflict detection and resolution. It is increasingly perceived that the present centralized way of managing traffic cannot scale up anymore. In order to deal with traffic growth, major research programs around the world, such as SESAR (Single European Sky ATM Research) and NextGen (US Next Generation Air Transportation System) consider automating some tasks previously done by controllers, allowing them to manage more flights simultaneously. In a more daring effort, decentralized flow management, whereby traffic flow management is delegated to individual aircraft, is also an option.

A. Air Traffic Management

The current air traffic system is structured [2] on a route network whose vertices are geographical positions called waypoints, through which an aircraft shall go. Each waypoint is identified by a name. Edges of the route network define the set of routes that can be followed by aircraft. Before departure, the pilot or the airline dispatcher is to submit a flight plan to the civil aviation authority, containing information related to the flight (route, departure and arrival airports, etc.). This flight route is defined by a set of waypoints identified by their names. Once airborne, however, modifications to the flight plan may be initiated by the flight crew or air traffic control, depending on local traffic and weather conditions.

The airspace of every country is usually divided into several sectors. Each sector is managed by a team of air traffic controllers in charge of planning trajectories, managing collision avoidance, and communicating with aircraft. Controllers can only take charge of a limited number of flights simultaneously. Sectors and route networks are designed in such a way that controllers' workload is reduced; they have only a limited number of flights to manage simultaneously in their sectors, each of them following a predefined route. They have to keep separation distances between each pair of aircraft above given threshold values: during cruise, a conflict occurs when two aircraft are separated by less than 5 nautical miles horizontally (5 NM = 9.26 km) and 1,000 feet vertically (1,000 ft = 304.8 m). When a conflict between two aircraft is foreseen, the controller requests that one or both pilots execute a maneuver, usually temporarily changing the heading or altitude to increase separation, before returning to initial flight path.

The air traffic is constantly increasing and the current air traffic structure is reaching its maximum capacity. To cope with this situation, parts of the control process could be delegated to automatic algorithms, like conflict detection and resolution or other traffic management tasks.

Aircraft are increasingly capable of communicating full estimated 4D trajectories over the next minutes by means of messages. These data can be transmitted by using Automatic Dependent Surveillance – Contract (ADS-C) [3, 2.2.6]. These messages are exchanged following a request-answer protocol. Usually, a ground station sends a request to a specific aircraft, which sends back a data frame containing aircraft identifier, position, speed and the predicted route composed of a set of 4D positions (3D + time). Information carried by ADS-C is more accurate than radar positioning since an aircraft uses GPS to get its position. Ground stations receiving these messages are able to provide controllers with accurate representation of air traffic. This is well adapted to airspace where radar coverage is not available (e.g. oceanic airspace).

Free flight is an air traffic management concept developed in the U.S. that enables aircraft to choose their path in low traffic zones with more freedom, by ignoring route networks. The implementation is currently studied by the NextGen research program. In Europe a similar concept was developed by the SESAR project; the Free Route Airspace is already deployed in some areas. These zones are managed or not by air traffic controllers. In the second alternative, automatic separation assurance systems would benefit from aircraft information exchange systems.

B. Conflicts Detection and Resolution Algorithms

Many studies have already been performed to design automatic conflicts resolution methods. Some are referenced in [4]. Most of those methods try to reproduce the way controllers regulate traffic by changing aircraft heading for a short period. According to air traffic controllers, those algorithms can interfere with their own decisions since the controllers and the algorithms take the same kind of decision in the same controlled areas and within the same time horizon [5].

In 2004, a new way to solve this problem was proposed as a part of the project ERASMUS (En Route Air traffic Soft Management Ultimate System). According to Villiers [5], instead of trying to reproduce what controllers are doing, those algorithms should help them by removing a part of the conflicts before they appear by slightly changing aircraft speed, for instance. This type of automated system organizes traffic in order to create a favorable traffic situation, more easily managed by controllers. Moreover, this algorithm does not interfere with controllers' decision process, since controllers mainly manage traffic in space dimensions by modifying aircraft heading, and the algorithm manage traffic in time dimension by regulating aircraft speed. This concept was validated with experiments [6] and human factors studies [7]. The speed regulation method was implemented as a genetic algorithm [6].

Since problems encountered in Air Traffic Management (ATM) are highly combinatorial, deterministic optimization methods tend to become inefficient when dealing with real traffic scenarios (involving hundreds or even thousands of aircraft). To overcome this situation, heuristics have been used in several research works, giving satisfactory results over a time horizon compatible with operational constraints [8].

Multi-agent systems can be used to develop heuristic algorithms, and have already been applied to air traffic management problems. Some studies focus on traffic regulation in free flight (or Free Route) zones. Aircraft flying in free flight areas must automatically be able to find conflict-free trajectories, respecting the required distance separation between aircraft. This problem can be solved using multi-agent systems, such as the one developed by Wollkind, Valasek and Ioerger [9], or the one developed by Sislak, Volf and Pechoucek [10]. Those algorithms solve conflicts using maneuvers such as heading and flight level changes. However, and as shown by Villiers, this regulation method can interfere with controllers' decision process.

The algorithm described in this paper aims at implementing the traffic regulation method of the project ERASMUS as a multi-agent system. As detailed in Section I-C, this type of algorithm has several advantages compared to global optimization methods used in ERASMUS.

C. Multi-Agent Systems

Multi-agent systems have been used to solve many problems in operations research, like regulation of urban transportation networks [11], design of mechanical systems [12], or path-finding problems [13]. This paradigm is often regarded as a kind of distributed artificial intelligence. Multi-agent systems are made of autonomous agents interacting among themselves and with their environment [14]. Usually, agents have a limited perception of environment and they partially know the internal state of their neighbors, via message exchanges. Their behavior can either be simple (whereby reactive agents are only influenced by environmental changes) or complex (whereby cognitive agents try to fulfill an objective).

Self-organization is a key aspect of multi-agent systems. Self-organization means that the system organization is not predefined, but emerges from interactions between agents, and gives a complex behavior to the overall system. If the rules that direct agents are carefully chosen, such a complex behavior can emerge at the system level from local interactions and behavior of agents. When multi-agent systems are used to solve operations research problems, a carefully chosen set of agent behaviors can help to find an overall solution to the problem (system level) by only using local rules (agents level).

Multi-agent systems can be implemented either within a computer simulation, or as a physical system that is composed of robots that are able to communicate and to interact among each other and with their environment. When agents are implemented within a computer simulation, computations of agents can be done in parallel, exploiting modern hardware architectures (multi-core processors, computations on graphic card). A multi-agent system can also run on a cluster of computers.

Those systems have several advantages compared to centralized decision algorithms. When correctly designed, they exhibit a good resilience when facing disruptive events [15].

Agents try to fulfill a goal and act in order to become closer to this objective. When they are confronted to local perturbations in their environment, they adapt their actions to take those changes into account, enabling the system to return to a new stable state. Since decisions are decentralized at agents level, the failure of an agent will not impact the whole system. Computer experiments indicate that, in general, when decisions are decentralized, a central regulation entity failure may prevent the system to work. In the field of information technologies, such a central point would be defined as a Single Point of Failure (SPOF).

D. Implementation of a Conflict Resolution Multi-agent System

Applied to air traffic management, the implementation of new onboard collaborative decision processes can be done progressively, whereby equipped aircraft cooperate among themselves and are given more freedom in their decisions than non-equipped aircraft (for example by constraining the latter to follow rigid corridors).

Even if current technology like ADS-C does not allow aircraft to really exchange data about trajectories directly from aircraft to aircraft, these data can be collected by ground stations. Traffic regulation algorithms like the one described in this article can then compute speed changes, which can then be sent back to aircraft. The distributed aspect of a multi-agent approach to the management of traffic is somewhat diminished since aircraft do not take decisions themselves. Yet, other advantages are preserved, like resilience and general performances.

This article describes a conflict resolution algorithm based on aircraft speed self-regulation. This implementation of the ERASMUS concept benefits from the use of multi-agent systems. Expected benefits include resilience to disruptive events like non-cooperative aircraft, and the ability to delegate the decision process onboard, supposing that aircraft will be able to exchange trajectory data among themselves. Section II details the algorithm. Section III describes a scenario based on actual traffic aiming at validating expected assets of this algorithm.

II. Aircraft Speed and Delay Self-Regulation for Conflict Resolution

A. Hypotheses

The algorithm described in this section regulates aircraft in cruise phase. Their altitude is supposed to be constant. Aircraft flying at different altitudes do not interact; they are never in conflict, and they ignore each other during the decision process.

Each flight has a preferred cruise speed depending on general aircraft performances and airlines preferences. Airlines can give priority to reducing fuel cost by reducing aircraft speed, or to the reduction of crew costs by increasing speed, which also increases fuel consumption. This setting is adjusted by using a value called Cost Index (CI), which is the ratio of the cost of flight time (including crew costs) to the cost of fuel.

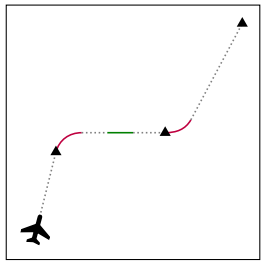

Figure 1. In the algorithm described in Section II, a trajectory is a curve composed of segments (constant speed in gray, acceleration in green) and of arcs of circles (in dark yellow). Aircraft flies from a waypoint to another (triangles).

Let v be the current speed of an aircraft and v_{opt} its optimal speed. In order to be inserted into a route network, aircraft may have to choose a speed v different from v_{opt} within a given interval. A lower bound $v_{min} = v_{opt} - 6\%$ allows a human or an algorithm to insert this aircraft into a flow without dramatically increasing fuel consumption [16]. A speed interval of $[v_{opt} - 6\%, v_{opt} + 3\%]$ is a common choice for speed regulation in en route airspace [17].

Aircraft acceleration and deceleration are fixed to $\pm 4{,}000\,\mathrm{NM/h^2}$ ($\pm 0.572\,\mathrm{m/s^2}$) for all aircraft. The standard turn rate [18, PCG S–6] of 3°/s is used so that a complete 360° turn is completed in 2 minutes.

In a general traffic situation or in free route scenarios, some conflicts cannot be solved by using speed control only, as in head-on encounters, for instance. The aim of this algorithm is to simplify the traffic by doing subliminal speed changes in order to help controllers. The speed regulation algorithm reduces the number of conflicts and delegates the remaining ones to air traffic controllers.

Thus, these conflicts need additional maneuvers to be solved. In this algorithm, aircraft are also able to delay departure time up to 10 minutes, in order to remove the remaining conflicts.

B. Algorithm

In this multi-agent system, aircraft are agents exchanging ADS-C messages containing estimated 4D trajectories. A trajectory is stored and exchanged as a sequence of arcs that can be straight segments (aircraft flying at a constant speed, accelerating, or decelerating) or arcs of circles (aircraft turning), as shown in Fig. 1. This curve is differentiable at least once everywhere.

This curve is built from a flight plan defined by a set of waypoints. Flight plan execution is as follows: an aircraft has to fly above each waypoint (Fig. 1). After an aircraft reaches a waypoint, it turns in order to head towards the next one. Speed changes are planed at given times (Section II-B2), and can occur only during the execution of straight trajectory segments.

In our model an aircraft cannot accelerate and turn at the same time. Thus, all turns can be represented by arcs of circles,

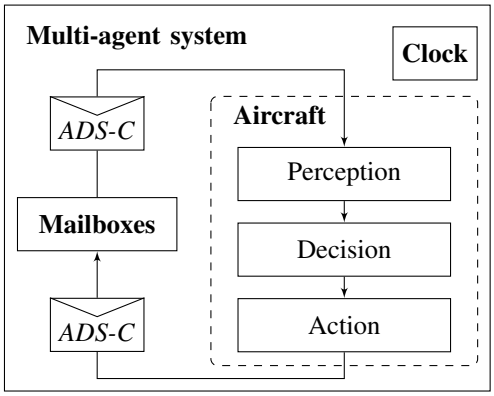

Figure 2. Multi-agent system lifecycle.

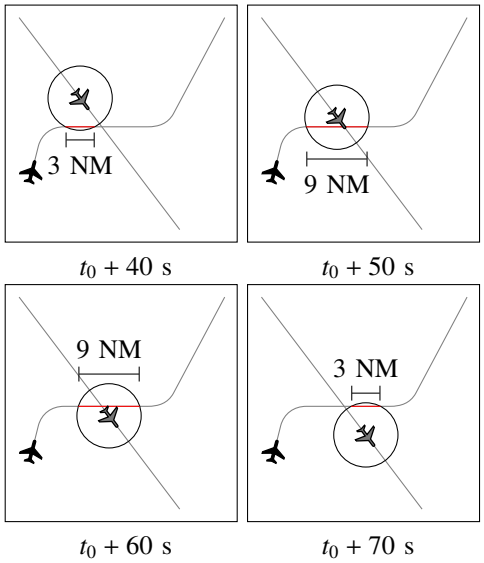

Figure 3. The black aircraft detects potential conflict with the gray aircraft for each time sample in the next minutes. A potential conflict is the intersection between a circle of 5 NM centered on the gray aircraft and path of the black aircraft (red segments).

enabling the computation of conflicts (Section II-B1a) with a simpler analytic method than if aircraft could accelerate during turns. Furthermore, flight plans are designed to minimize flown distance. Heading changes are usually small, and turns rarely last more than on the order of tens of seconds. Thus, postponing speed changes after turns has little or no influence on the decision process.

The multi-agent system (Fig. 2) is timed by a global clock. Each tick corresponds to one second in the simulation. All agents are synchronized: at the end of an iteration, agents drop off messages into the mailbox. When the next iteration begins, those messages are delivered to addressee agents. Therefore, even if agents processes are run asynchronously, agents work logically in parallel. This choice helps the overall system to avoid problems related to sequence order. For performance reasons, agents are run in parallel, using multiple threads.

During the lifecycle of an agent, a sequence of three steps is repeated at every iteration of the multi-agent system until the agent is removed. The perception step allows agents to receive ADS-C messages and refresh their internal representation of the airspace. During the decision step, aircraft plan speed changes on the basis of this internal representation. In the action step, agents update their position using updated speed, and broadcast an ADS-C message. Aircraft lifecycle steps are detailed below.

1) Perception: Each aircraft agent first receives messages from its neighbors, from which it extracts 4D trajectories. In order to detect conflicts, the aircraft then samples other aircraft trajectories to get their predicted position every 10 seconds during 20 minutes (Fig. 3).

a) Conflict Detection: This algorithm aims only to regulate aircraft trajectories in time dimension. The space and time dimension are handled separately by the algorithm. The spatial dimensions of a trajectory are the dimensions of the geometric space spanned by the trajectory in 3D. Altitude of a trajectory is supposed to be constant, and aircraft vertically separated will not be considered in conflict. The temporal dimension is defined by aircraft speeds at each instant. The algorithm regulates only the time dimension. The internal representation of an aircraft trajectory is the distance flown by the aircraft as a function of the time, and can be represented as a curve in a 2D space, as in Fig. 4.

In order to compute speed changes, conflicts have to be represented in a form understandable by the decision process. Conflicts are also projected into the 2D space where the aircraft trajectory is already projected (Fig. 4).

A conflict occurs when the distance between two aircraft flying at the same altitude becomes smaller than 5 NM on a horizontal level. When an aircraft follows a path, its future positions are defined by its own current position and by its speed changes. Since speed is to be chosen during the decision phase, all potential conflicts with the aircraft's neighbors need to be detected, regardless of its own speed. Then, for each intruder predicted position, the algorithm searches for possible intersections between its own path and a circle of 5 NM centered on the intruder predicted position (Fig. 3). Thus, whatever speed changes the aircraft will choose, it will always manage to detect conflicts.

b) Internal Representation of Own Trajectory: Since an aircraft strictly follows a fixed geometrical path, the decision process only modifies speed. As previously mentioned, to manipulate a trajectory defined by a 4D curve is not necessary: this 4D trajectory can be simplified by using the traveled distance over the route as a function of time, by integrating the instantaneous speed over time. In other words, the aircraft trajectory can be represented as a 2D curve defined by the arc length as a function of time. This curve is then projected into a 2D space (Fig. 4).

Conflicts detected during the conflict detection step (Section II-B1a) are also projected in this space. Each potential intrusion extracted from the sampled neighbor's trajectory is projected according to its position along the path (Fig. 3). The portion of the path intersecting with the circle of 5 NM cen-

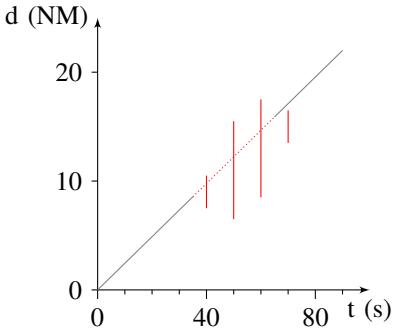

Figure 4. Representation of the 2D trajectory of the black aircraft (distance as a function of time) and intrusions of the gray neighbor in the black aircraft's path at each time sample (vertical red segments) of the Fig. 3. A conflict will occur if the black aircraft maintains its initial speed: the curve representing the black aircraft's trajectory intersects with the vertical segments that represents potential conflicts.

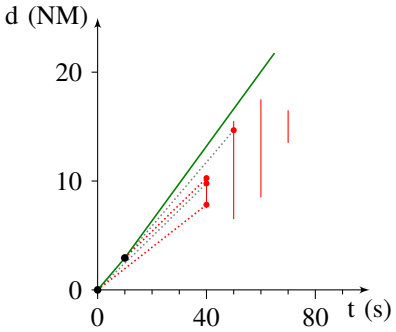

Figure 5. Exploration of the decision tree. For each time step, 3 choices are tested: to accelerate (green), to cruise (gray), to decelerate (blue). The choice leading to the longest conflict-free trajectory is applied. In this case, it accelerates two times at the beginning, then it cruises.

tered on the intruder's position gives an interval of positions forbidden to the agent (Fig. 4).

An example of this process is shown in Fig. 3 and 4. In Fig. 3, conflicts are symbolized by red segments representing intersection of 5 NM-radius circles centered on intruder aircraft and of the spatial components of aircraft's own trajectory. In Fig. 4, these red segments are projected in the 2D internal representation of the trajectory.

2) Decision: Decisions are updated every 8 seconds. The instant of the first decision is randomly chosen within the first 8 seconds of agent's lifetime. It means that between two executions of the same traffic scenario, aircraft will take decisions in a different order, and decisions of the first aircraft has an influence over decision of the next ones.

Using conflicts projected in the 2D representation (displayed in Fig. 4), aircraft can plan speed changes. The goal is to avoid conflicts by way of speed changes. This goal is achieved by exploring a decision tree where for each time sample, the aircraft can maintain its speed, accelerate or decelerate.

A time step of 5 seconds was chosen. Aircraft acceleration rate is ±4,000 NM/h, or ±1.11 NM/s (Section II-A). An aircraft whose optimal speed v_{opt} is 447 kt has an admissible

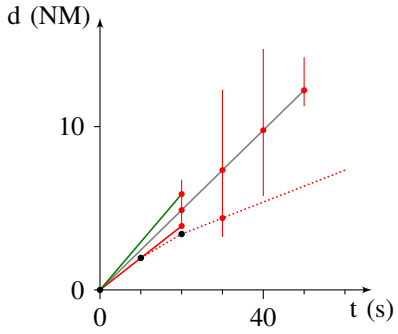

Figure 6. The conflict is unsolvable because of the objective function that tries to maximize the delay before the first conflict; in the first time sample, all the possible choices (to accelerate, to cruise or to decelerate) lead to a conflict at t = 20 s. In this case, the aircraft chooses to cruise, which leads to a severe conflict lasting 40 seconds. An alternative objective function that minimizes the duration of a conflict can choose to decelerate three times in order to cause a less severe conflict lasting only 10 s (dashed segments).

speed range of [420 kt, 460 kt], so it takes 36 seconds to accelerate from its minimum to its maximum speed. Aircraft accelerate constantly between two time steps. Thus, a time step of 5 seconds means that aircraft can choose 7 different speeds. Experiments (Section III) shown that it is sufficient to solve all the conflicts.

For each time sample, three choices are tested: cruise, acceleration or deceleration at the maximum rate (Fig. 5). Then at each time step and for each possible decision, the time of the first conflict is computed. The problem is solved using a greedy algorithm that locally maximizes the time of the first conflict.

A pseudo-code describing the decision-making process of each aircraft is presented in Appendix A. In this process, the set of decisions D is iteratively constructed. For each time step, acceleration (acc), deceleration (dec) and cruise (cr) choices are tested: the time when the first conflict occurs is stored in the variable cflTime. Then, the algorithm looks for the decision leading to the latest conflict date. If speed constraints are respected (checked by the function isValid()), this decision is accepted.

Using a greedy algorithm makes possible to get good results (Section III) after a short period of computation. Yet, since a greedy algorithm is only a local optimization process, it may find a local optimum and be unable to find a conflict-free solution, even if one exists as shown in Fig. 6. Therefore, some conflicts cannot be solved. The choice of this method is a compromise between the computation time and the quality of the results.

Results given by the greedy algorithm can nonetheless be refined by the addition of intermediate decisions at each time step (e.g. accelerations and decelerations of ±2,000 NM/h^2 and ±4,000 NM/h^2), but at the cost of longer computation times.

3) Action: In the action step, the set of decisions D computed by the algorithm of Section II-B2 is used to generate the new 4D trajectory, which is communicated to neighbors

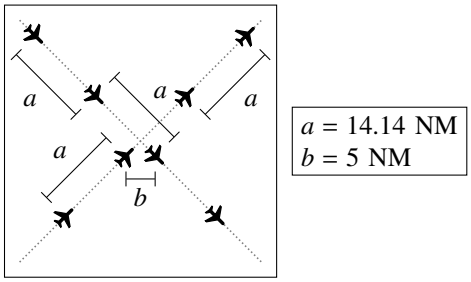

Figure 7. Intersecting flows of aircraft regulated by the multi-agent system. At a perpendicular intersection, if aircraft flying at the same speed are separated by 14.14 NM on each route, and if they alternately cross the intersection, they can keep a minimum distance of 5 NM.

using an ADS-C message.

4) Delay of departure time: Aircraft are generated 10 minutes before departure. They immediately start to send ADS-C messages containing their 4D trajectory. This behavior allows aircraft to plan speed changes before actually appearing in the simulation. It also enables aircraft already in flight to anticipate their presence enroute.

An additional action is possible by delaying departure times. Aircraft can delay their departure time up to 10 minutes. As in the pseudo-code in Appendix B, agents call iteratively the speed regulation method described in Section II-B2 and in Appendix A with increasing delay values. The aim of this function is to maximize the date of the first conflict that cannot be solved by the speed regulation algorithm. Aircraft can adjust departure delay during the decision step (Section II-B2) until the departure time. The latest allowed departure time is 10 minutes after the departure time initially set in the flight plan.

Delay values start from 0 to 10 min. In order to reduce the number of iterations, the time step Δt is set to 1 s when delay is lower than 15 s, Δt is set to 5 s when delay is lower than 3 min, otherwise Δt is set to 10 s. The tested delays are then $\{0, 1, 2, \ldots, 14, 15, 20, \ldots, 175, 180, 190, \ldots, 600\}$. Then, the algorithm iterates 90 times instead of 600 times if the time step was fixed to 1 s from 0 to 10 min. The process stops if a conflict-free solution is found.

Since all agents apply the same decision process iteratively, numerical experiments indicate that the system converges to a stable state in which conflicts are solved. When aircraft follow the same route, they fly at similar speeds in order to avoid conflicts. When they follow intersecting routes, the separation distance between aircraft of each route becomes similar in order to allow them to cross alternatively the intersection point. The Figure 7 shows an example of two perpendicular flows of aircraft. In order to maintain a minimum distance of 5 NM, aircraft of each flow have to be separated by:

$$2 \cdot 5\sqrt{2} = 14.14 \text{ NM} \quad (1)$$

and cross the intersection alternately.

In order to validate the algorithm, various traffic scenarios were tested. A description of the scenarios and some results can be found in Section III.

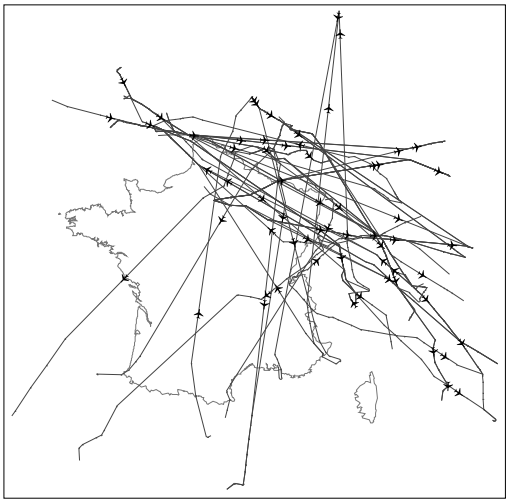

Figure 8. Traffic scenario using actual flight plans, described in Section II. This picture displays flight plans of aircraft flying at 12:00.

III. Test Scenarios Based on Actual Flight Plans

In order to validate the algorithm, different scenarios based on actual traffic were tested. Scenarios are based on flight plans of aircraft flying over France during 10 hours (from 4 AM to 2 PM), as shown in Fig. 8. Only the flights at 37,000 ft, one of the most densely occupied altitude, have been analyzed. 283 aircraft fly at this altitude in this scenario.

Three sets of scenarios were tested (Section III-A). In the first set, aircraft regulate speed but do not change departure time. In the second set of scenarios, aircraft can delay departure time up to ten minutes, but do not regulate speed and fly at their optimal speed. In the third set, aircraft regulate departure delay and speed.

In order to test resilience of the algorithm to disruptive events, for each set of scenarios, the decision process was disabled for a given subset of aircraft (Section III-B). Percentages of non-cooperative aircraft were chosen to be 10 %, 20 %, 50 % and 80 %.

During simulations, the distances between all pairs of aircraft were monitored. Each time aircraft were separated by less than 5 NM, a conflict was registered (a single record for the duration of the conflict).

Each scenario was run 10 times. The number of conflicts was recorded (Fig. 9 and 11), as well as the number of accelerations (Fig. 12) and the delay (Fig. 13) chosen by every cooperative aircraft.

Results vary between runs of the same scenario. In Fig. 9, the minimum and maximum measured number of conflicts are indicated alongside the average number. These variations have two origins. Firstly, aircraft update their decisions every 8 seconds, but the instant of the first decision is randomly chosen within the first seconds of agents' lifetime. It means that from one run to the other, aircraft will take decisions in a different order, and decisions of the first aircraft has an influence over decision of the next ones. Secondly, in scenarios with non-cooperative aircraft, these aircraft are randomly cho-

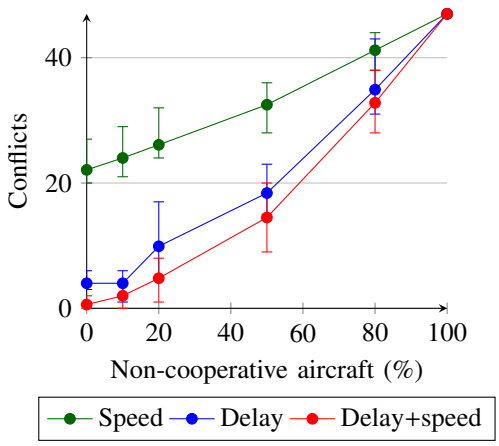

Figure 9. Number of conflicts as a function of the percentage of non-cooperative aircraft. Results are given for each set of scenarios: with speed regulation only, with delay regulation only, and with both methods enabled. Each scenario was run 10 times; the minimum, average and maximum number of conflicts was measured.

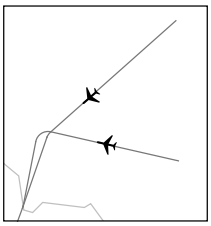

Figure 10. When aircraft cannot delay their departure time, they are sometimes starting at too close positions and speed regulation is not sufficient to solve these conflicts.

sen within the set of agents for each simulation run. An aircraft can cooperate in a run and be non-cooperative in another one. The influence of an aircraft on the overall traffic can vary, depending for instance if its trajectory crosses congested areas or not.

In order to provide a baseline, a first simulation was run in which decision process was disabled for all aircraft. This scenario generated 47 conflicts. In Section III-A we will discuss the algorithm's ability to reduce conflicts when aircraft cooperate against the baseline. In Section III-B we will discuss the algorithm resilience when confronted to disrupted events.

A. 100 % of Cooperative Aircraft

In this section, we discuss the algorithm's ability to reduce conflicts when all aircraft cooperate. In the first set of scenarios, aircraft only regulate speed. As shown in Fig. 9, when all aircraft cooperate, 24.9 of the 47 initial conflicts are solved on average (−53 %). Solving conflicts requires that each aircraft accelerate 7.2 times on average, for an accumulated acceleration duration of 35.9 seconds (Fig. 12).

22.1 conflicts are not solvable by the speed regulation algorithm alone. As shown in Fig. 10, some aircraft start from positions that lead to an unsolvable conflict. Furthermore, speed regulation alone cannot solve face-to-face conflicts.

In the second set of scenarios, aircraft do not regulate speed, but they are able to delay departure time up to 10 minutes. This strategy allows the system to remove conflicts that cannot be solved by speed regulation only, such as the conflict depicted in Fig. 10. The number of conflicts is then much lower than for the previous set of scenarios, with only 4 unsolved conflicts on average (−91.5 %). Aircraft delay their departure by 13.3 seconds on average, as displayed in Fig. 13.

In the last set of scenarios, aircraft can regulate speed and departure time. This is the most favorable situation, since the degree of decision freedom is the greatest. Aircraft can solve conflicts such as the one shown in Fig. 10. When all aircraft cooperate, between 0 and 2 conflict remain, with an average of 0.6 unsolved conflicts.

Aircraft are able to solve conflicts with 5.8 speed changes on average, for an accumulated acceleration duration of 28.9 seconds. Comparing with the first set of scenarios (aircraft regulate speed only), allowing aircraft to delay departure helps them to slightly reduce the number of speed changes needed to solve conflicts. The average delay is also reduced by 56.6 % compared to the second set of scenarios (aircraft only delay departure time); when all aircraft are cooperative, the average delays decreases from 13.3 seconds to 5.8 seconds.

B. Resilience

In order to validate the algorithm's resilience, for each of the three sets of scenarios (speed regulation only, departure delay only, speed and delay regulation enabled), 4 additional scenarios were tested with different percentages of non-cooperative aircraft. In these scenarios, the ability to make decisions was removed for 10, 20, 50 and 80 % of agents. The cooperative aircraft consider the non-cooperative ones as constraints on their decision process, and solve conflicts if admissible delay and speed interval allow them to do so. In these simulations, resilience was analyzed at three different levels. At the level of all agents, resilience is related to air traffic controllers' workload. At the level of cooperative agents, resilience is related to overall performances of the algorithm. And at the level of a single agent, resilience is related to the quality of aircraft decisions.

1) Resilience at the Level of All Agents: Air traffic controllers' workload is partially related to the complexity of a traffic situation. A part of this complexity comes from the number of conflicts controllers have to solve. Fig. 9 displays the total number of conflicts that could not be solved by the algorithm as a function of the number of non-cooperative aircraft. These residual conflicts must be solved by controllers, and have an influence over traffic complexity. Then, values displayed in Fig. 9 gives a trend of the perceived complexity as a function of the ratio of non-cooperative aircraft. When 20 % of aircraft do not cooperate, the number of conflicts increases by about 10 % with respect to when all aircraft cooperate. In the first set of scenarios (only speed regulation), the number of conflicts increases by 8.5 %, from 22.1 to 26.1 conflicts over the 47 initial ones. In the second set (only departure delays allowed), the number of conflicts increases by 12.6 %, from 4 to 9.9 conflicts compared with the initial number of conflicts.

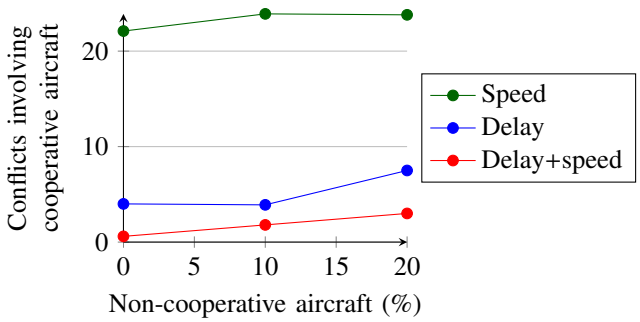

Figure 11. Number of conflicts involving cooperative aircraft as a function of the percentage of non-cooperative aircraft. The number of conflicts is lower than in Fig. 9; conflicts between non-cooperative aircraft are not counted in this plot because they are not significant to study the algorithm performances. Conflicts counted in this Figure imply at least one cooperative aircraft. Results are given for each set of scenarios: with speed regulation only, with delay regulation only, and with both methods enabled.

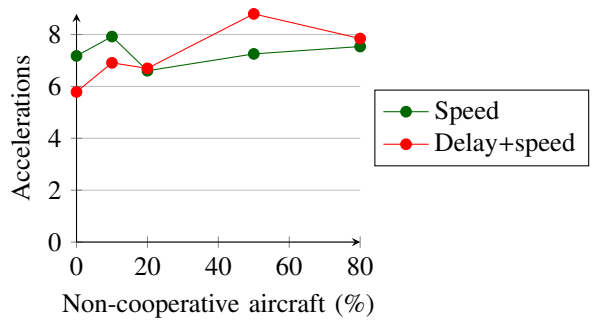

Figure 12. Average number of accelerations decided by cooperative aircraft in function of the percentage of non-cooperative aircraft. Results are given with delay regulation enabled and disabled.

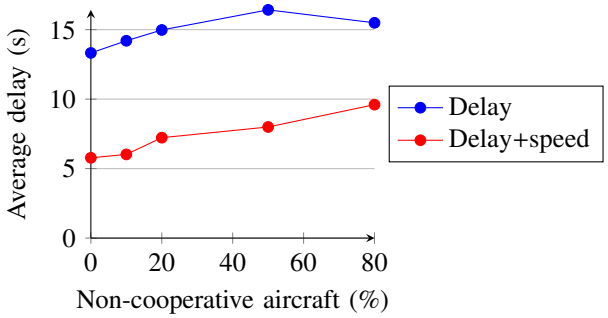

Figure 13. Average delay decided by cooperative aircraft in function of the percentage of non-cooperative aircraft. Results are given with speed regulation enabled and disabled.

In the last set (delay and speed regulation), the number of conflicts increases by 8.9 %, from 0.6 to 4.8 conflicts compared to the 47 initial ones.

Disabling the decision-making ability for 20 % of aircraft increases the number of conflicts only by 10 % on average in comparison to when all aircraft cooperate. This value indicates that the increase of controllers' workload caused by the number of conflicts will be limited if some aircraft do not cooperate.

2) Resilience at the Level of Cooperative Agents: The number of conflicts increases with the number of non-cooperative aircraft. These additional conflicts can be categorized in two groups: conflicts between two non-cooperative aircraft, and conflicts involving at least one cooperative aircraft. The first group has to be taken into account in order to monitor overall performances of the system, but does not provide much information about performances of the decision process. These conflicts cannot be solved, and their number tends to 47 as the ratio of cooperative aircraft tends to 0.

The number of conflicts involving at least one cooperative aircraft provides more information about the algorithm performances, because these conflicts can be solved by the decision process. As shown in Fig. 11, the number of conflicts belonging to this category slightly increases with the number of non-cooperative aircraft, by 5.4 % on average while 20 % of aircraft do not cooperate. In the scenario where only speed is regulated, the number of conflicts increases by 3.6 %, from 22.1 to 23.8 conflicts over the 47 initial ones. In the scenario where only delay is applied, conflicts increase by 7.4 %, from 4 to 7.5 conflicts. When speed regulation and delay are applied, the number of conflicts increase by 5.1 %, from 0.6 to 3 conflicts.

In other words, half of the conflicts appearing while 20 % of aircraft do not cooperate are caused only by non-cooperative aircraft. When one fifth of aircraft do not contribute to conflicts resolution, the number of conflicts remains stable with an increase of 5.4 % of conflicts compared to when all aircraft cooperate. This value indicates that the algorithm is resilient to disruptive events such as one aircraft being, or becoming, non cooperative.

3) Resilience at the Level of a Single Agent: At the level of individual agents, the number of speed changes and average delay chosen by each aircraft was also monitored for each scenario. The average values in function of the ratio of non-cooperative aircraft are shown in Fig. 12 and Fig. 13. The average number of accelerations is stable as the number of non-cooperative aircraft increases, and is bounded between 5.8 and 8.8 speed changes for all scenarios.

Delay slightly increases with the number of non-cooperative aircraft. The maximum delay increase is 3.8 seconds while the number of non-cooperative aircraft increases from 0 to 80 %, increasing from 5.7 s to 9.6 s when delay and speed regulation are applied by aircraft.

The average number of speed changes and delay before departure are stable while the number of non-cooperative aircraft increases. This indicates that the aircraft decision process is resilient against an important number of non-cooperative aircraft.

In conclusion, the multi-agent system is resilient at three different scales. At system level, the limited increase of the number of conflicts means a limited increase of air traffic controllers' workload if some aircraft do not cooperate to solve conflicts. If conflicts involving only non-cooperative aircraft are ignored, increase of the number of conflicts is limited, indicating resilience of the algorithm. Finally, the number of

accelerations and the average delay remain stable, indicating resilience at the level of individual agents.

IV. Conclusion

The algorithm described in this article has several advantages related to the usage of multi-agent systems. It is able to solve up to 100 % of the conflicts in the tested scenario, is resilient to perturbations like non-cooperative agents, and could eventually be implemented on board, removing the need to rely on ground equipment.

Speed regulation alone is not able to solve all conflicts, and is not meant to do so. The algorithm described in this article is designed to assist air traffic controllers, and not to replace them. To successfully solve all conflicts, aircraft have to execute other types of maneuvers in addition to speed regulation, like delay of the departure time, flight level changes or heading changes. Allowing aircraft to delay departure time helps the algorithm to remove all the remaining conflicts. Nowadays, these decisions are taken by air traffic controllers and are not planned to be delegated to algorithms in human-controlled airspace.

Our next research will aim at implementing a global optimization method to solve the problem described in this article. It will allow to measure the difference in terms of computation time and result optimality between results returned by this multi-agent system and by a global optimization method, often used to solve similar problems.

Thanks to the ability of multi-agent systems to integrate non-cooperative agents and to recover from disruptive events, they offer a good framework to mix human-controlled traffic with an automated one. Eventually, these algorithms could then collaborate with humans in air traffic management applications, because they work over different scales, time-wise for the algorithm and space-wise for humans.

Appendix A
Speed Regulation Algorithm

function OptimizeSpeed(delay)
  choices ← {cr, acc, dec}
  $D \leftarrow \{\}$
  for $t \leftarrow t_0 + \text{delay}$ **to** t_{end} **step** Δt **do**
    bestChoice ← cr
    bestCflTime ← t
    for c **in** choices **do**
      cflTime ← getFirstConflict($D \cup \{c\}$)
      if isValid($D \cup \{c\}$) **then**
        if cflTime > bestCflTime **then**
          bestChoice ← c
          bestCflTime ← cflTime
        end if
      end if
    end for
    $D \leftarrow D \cup \{\text{bestChoice}\}$
  end for
  return D
end function

Appendix B
Delay Selection Algorithm

function OptimizeDelay
  possibleDelays = $\{0, 1, 2, \ldots, 14, \} \cup$
    $\{15, 20, \ldots, 175\} \cup \{180, 190, \ldots, 600\}$
  bestDelay = 0
  bestScore = 0
  for delay **in** possibleDelays **do**
    score ← OptimizeSpeed(delay)
    if score > bestScore **then**
      bestDelay = delay
    end if
    if score = maxScore **then**
      return bestDelay
    end if
  end for
  return bestDelay
end function

References

[1] International Civil Aviation Organization, *Global Air Transport Outlook to 2030 and Trends to 2040*, ser. ICAO circular. International Civil Aviation Organization, 2013.

[2] D. Isaacson, A. Sadovski, and D. Davis, "Tactical scheduling for precision air traffic operations: Past research and current problems," *Journal of Aerospace Information Systems*, vol. 11, no. 4, pp. 234–257, 2014.

[3] *Global Operational Data Link Document (GOLD)*, http://www.icao.int/APAC/Documents/edocs/GOLD_2Edition.pdf, International Civil Aviation Organization, apr 2013.

[4] J. Kuchar and L. Yang, "A review of conflict detection and resolution modeling methods," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, no. 4, pp. 179–189, Dec 2000.

[5] J. Villiers, *Automatisation du contrôle de la circulation aérienne: "ERASMUS", une voie conviviale pour franchir le mur de la capacité*, ser. Etudes & documents - I.T.A. ITA, 2004. [Online]. Available: https://books.google.fr/books?id=FXuFQwAACAAJ

[6] D. Bonini, C. Dupré, and G. Granger, "How ERASMUS can support an increase in capacity in 2020," in *Proceedings of the 7th International Conference on Computing, Communications and Control Technologies: CCCT*, 2009.

[7] G. Chaloulos, E. Crück, and J. Lygeros, "A simulation based study of subliminal control for air traffic management," *Transportation research part C: Emerging technologies*, vol. 18, no. 6, pp. 963–974, 2010.

[8] D. Delahaye, N. Durand, J.-M. Alliot, and M. Schoenauer, "Genetic algorithms for Air Traffic Control systems," in *IFORS 1996, 14th Triennal Conference of the International Federation of Operational Research Sociaties*, Vancouver, Canada, Jul. 1996. [Online]. Available: https://hal-enac.archives-ouvertes.fr/hal-00938864

[9] S. Wollkind, J. Valasek, and T. R. Ioerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *AIAA guidance, navigation, and control conference*, 2004, pp. 1–11.

[10] D. Sislak, P. Volf, and M. Pechoucek, "Agent-based cooperative decentralized airplane-collision avoidance," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 36–46, March 2011.

[11] F. Balbo and S. Pinson, "An agent oriented approach to transportation regulation support systems," in *Proceedings of the 5th Workshop in Agent in Traffic and Transport*, 2008, pp. 225–242.

[12] D. Capera, M.-P. Gleizes, and P. Glize, "Self-organizing agents for mechanical design," in *Engineering Self-Organising Systems*. Springer, 2004, pp. 169–185.

[13] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, Nov 2006.

[14] J. Ferber, *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley Reading, 1999, vol. 1.

[15] C. Rieger, K. Moore, and T. Baldwin, "Resilient control systems: A multi-agent dynamic systems perspective," in *Electro/Information Technology (EIT), 2013 IEEE International Conference on*, May 2013, pp. 1–16.

[16] L. Delgado and X. Prats, "Fuel consumption assessment for speed variation concepts during the cruise phase," in *Proceedings of the Conference on Air Traffic Management (ATM) Economics*, 2009.

[17] P. Averty, B. Johansson, J. Wise, and C. Capsie, "Could ERASMUS speed adjustments be identifiable by air traffic controllers?" in *Proceedings of the VII USA/Europe Air Traffic Management R&D seminar, Barcelona*, jul 2007.

[18] *Aeronautical Information Manual, Official Guide to Basic Flight Information and ATC Procedures*, http://www.faa.gov/atpubs, Federal Aviation Administration, apr 2014.