Multi-agent Systems to Help Managing Air Traffic Structure

Romaric Breil, Daniel Delahaye, Laurent Lapasset, Eric Féron

Multi-agent Systems to Help Managing Air Traffic Structure

Romaric Breil*[1 2], Daniel Delahaye[1], Laurent Lapasset[1], and Éric Féron[3]

[1]Université Fédérale de Toulouse,
École Nationale de l'Aviation Civile,
Toulouse, France
[2]Capgemini Technology Services, Toulouse, France
[3]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 in the time dimension (e.g. sequencing and merging of aircraft flows taking off or landing at airports).

The objective of this paper is to develop a methodology that allows the traffic to self-organize in the time and space dimensions 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 which aircraft cooperate to structure traffic. Multi-agent systems have several advantages, including a good resilience when confronted with disruptive events.

In this system, three algorithms have been implemented, aiming at reducing traffic complexity in three different ways. The first algorithm allows aircraft agents flying on a route network to regulate speed in order to reduce the number of conflicts, a conflict occurring when two aircraft do not respect separation norms. The second algorithm allows aircraft to solve conflicts when the traffic is not structured by a route network. The third algorithm creates temporary local route networks allowing to structure traffic.

The three algorithms implemented in this multi-agent system allow to decrease overall traffic complexity, which becomes easier to manage by air traffic controllers. This algorithm was applied on realistic examples and was able to structure traffic in a resilient way.

*Corresponding author: romaric.breil@recherche.enac.fr

Keywords: Air traffic management, multi-agent system, conflict resolution, routes topology, complexity minimization.

1 Introduction

Air traffic volume has been constantly increasing during the past decades, and ICAO (International Civil Aviation Organization) predicts [25] 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. In doing so, required separation distance is maintained between all aircraft. This task is known as conflict avoidance. 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, like 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 at the same time. In a more daring effort, decentralized flow management, whereby traffic flow management is delegated to individual aircraft, is also an option.

1.1 Air Traffic Management

Air Traffic Management (ATM) includes two functions: Air Traffic Flow Management (ATFM) which deals with flow control and Air Traffic Control (ATC) which deals with separation assurance [24]. Air traffic is highly structured [27]. Aircraft cruise is controlled by Area Control Centers (ACC), also known in the U.S. as Air Route Traffic Control Center (ARTCC) [17, PCG A–12]. The airspace of every country is usually divided into several regions, each one controlled by an ACC. ACCs are here again partitioned into several subregions called sectors. Each sector is regulated by a team of air traffic controllers. Each team is usually composed of two controllers: one is in charge of communicating with aircraft, while the other is in charge of trajectory planning and collision avoidance management.

Before departure, the pilot or the airline dispatcher is to submit a document called flight plan to the civil aviation authority. The flight plan contains much information, including its departure and arrival airports, departure time, and the flight path. This flight path is defined by a set of geographical positions, called waypoints, through which an aircraft shall go. Each waypoint is identified by a name. Air traffic is structured on a route network whose vertices are the waypoints. 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.

Sectors and route networks are designed in such a way that air traffic controllers' workload is reduced. In addition, Air Traffic Flow Management procedures ensure that air traffic demand does not exceed sectors capacity. Thus,

air traffic controllers have only a limited number of flights to regulate simultaneously in their sectors, each of them following a predefined path. They will then have to keep separation distances between each pair of aircraft above given threshold values: in cruise phase, a conflict occurs when two aircraft are separated by less than 5 nautical miles (1 NM = 1,852 km) horizontally and 1,000 feet vertically (1 ft = 30.48 cm). When a conflict is detected (or predicted to occur soon), the controller requests that one or both pilots execute a maneuver, usually temporarily changing the heading to increase separation, before returning to initial flight path.

To allow aircraft position and speed information exchange, more and more aircraft are equipped with a system called Automatic Dependent Surveillance-Broadcast (ADS-B) [17, 4–5–7]. This system periodically broadcasts messages containing aircraft identifier, position and speed (ADS-B Out). Ground stations receive these messages and are able to provide controllers with accurate representation of air traffic. Aircraft carrying an ADS-B In system enable airborne traffic situational awareness.

Information carried by ADS-B is more accurate than radar positioning since aircraft uses Global Positioning System (GPS) to get its position. Yet, for the moment, only a fraction of the air fleet is equipped; that is why ADS-B can only be used as a radar complement, but it cannot replace it. However, ADS-B Out will become mandatory in the controlled airspace of several countries; for instance, European Union aircraft will have to be equipped with ADS-B Out in 2017 [15] so will the U.S. in 2020 [16].

Air traffic controllers can only take charge of a limited number of flights simultaneously and the current structure of air traffic is reaching its maximum capacity. To cope with this situation, parts of the control process could be delegated to algorithms that manage conflicts detection and resolution or other traffic management tasks.

Some areas around the world cannot be covered with radar antennas such as deserts and oceans; this explains why air traffic controllers regulate those areas with difficulty. Consequently, for safety reasons, large time separations are imposed between aircraft entering those areas (several minutes or tens of minutes), so as to ensure the separation distance minimum is respected. Systems allowing aircraft information exchange (like ADS-B) are currently deployed and will eventually allow a better traffic management in those areas [34]. Automatic separation assurance systems could benefit from this improvement.

Free Route Airspace [14] and Free Flight [23] are two air traffic management procedures (developed respectively in Europe and U.S.) that enable aircraft to choose their path in low traffic zones more freely by ignoring route networks. These zones are regulated or not by air traffic controllers. Both SESAR and NextGen research programs are currently studying the implementation of those concepts in their respective airspaces. In those areas, more automation will also be needed to ensure conflict-free trajectories.

Many studies have already been done to make automatic conflicts resolution (some are referenced in [28]). Since problems encountered in Air Traffic Management (ATM) are highly combinatorial, classic optimization algorithms tend

to become inefficient when dealing with real traffic scenarios (from hundreds to thousands of aircraft). To overcome this situation, heuristics have been used in several research works, giving approximate but good results in a length of time compatible with operational constraints [9].

1.2 Multi-Agent Systems

Multi-agent systems have been used to solve many problems in operations research, like regulation of urban transportation networks [3], design of mechanical systems [6], or path-finding algorithm [12]. 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 [19]. 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 (by which reactive agents are only influenced by environmental changes) or complex (by which cognitive agents try to fulfill an objective).

Self-organization is a key aspect of multi-agent systems. If the rules that direct agents are carefully chosen, 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 which are able to communicate and to interact with their environment.

Those systems have several advantages compared to centralized decision methods. Firstly, when correctly designed, they exhibit a good resilience when confronted with disruptive events [33]. Agents try to fulfill a goal and act in order to become closer to this objective. When they are confronted with local perturbations in their environment, they adapt their actions to take those changes into account, enabling the system to get back to a new stable state.

Secondly, decisions are decentralized at agents level, and the failure of an agent will not compromise the whole system. In centralized decision processes, 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).

When agents are implemented within a computer simulation, computations of agents can be done in parallel, exploiting modern hardware architecture (multi-core processors, computations on graphic card). A multi-agent system can also run on a cluster of computers.

Lastly, applied to air traffic management, 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).

Multi-agent systems have already been applied to air traffic management problems. Canino *et al.* [5] simulate air traffic control to validate the technical improvements in information sharing and communications between aircraft and area control centers newly introduced by SESAR and NextGen research programs. Pritchett *et al.* [30] use a multi-agent system to evaluate safety issues in air traffic management. Feigh *et al.* [18] simulates the air transportation system and study the emergence of complex behaviors at the system level.

Other projects focus on traffic regulation in Free Flight (or Free Route) zones. Aircraft flying in Free Flight areas must be able to find by themselves conflict-free trajectories, respecting the required distance separation between aircraft. This problem can be solved by multi-agent systems, such as the one that was developed by Wollkind, Valasek and Ioerger [38], or the one developed by Sislak, Volf and Pechoucek [36].

This article shows how multi-agent systems can be used to address various problems in air traffic management. Section 2 describes a conflict resolution algorithm based on aircraft speed self-regulation for traffic evolving on a network. Section 3 puts forward a broader approach to conflicts resolution by speed regulation, for unstructured traffic. Section 4 describes an on-demand local route network modification algorithm aiming at reducing local peaks of traffic complexity.

2 Aircraft Speed Self-Regulation in Miles-in-Trail Traffic

Our first experiment about multi-agent systems is related to the management of an intersection of two Miles-in-Trail (MIT) traffic flows. Miles-in-Trail [21] is a method used by U.S. controllers to reduce air traffic complexity. When flight density increases in an area, traffic is structured into flows of aircraft following the same path. Flights are separated by a given distance (for example 20 NM). This manner of structuring forms queues of aircraft. Those queues are easier to perceive and to manage by controllers. Their job is then to monitor inter-aircraft spacing and to apply speed control whenever it appears to be necessary.

Miles-in-Trail also makes merging or crossing of flows easier. Aircraft must be separated by at least 5 NM, which defines the maximum aircraft density in a flow. Therefore, when two flows merge, each one must apply a 10 NM Miles-in-Trail so that aircraft alternatively come from the first and the second flow. The same reasoning is applied to crossing flows (Figure 1) where aircraft must be separated by more than 10 NM, depending on the angle between flows (Section 2.3.3).

2.1 Comparison with a Similar Method in the Literature

The algorithm described in this Section is similar to a method developed by Chipalkatty *et al.* (2012) [8]. In that paper, aircraft speed and arrival date are regulated in a terminal maneuvering area, on a route network where aircraft

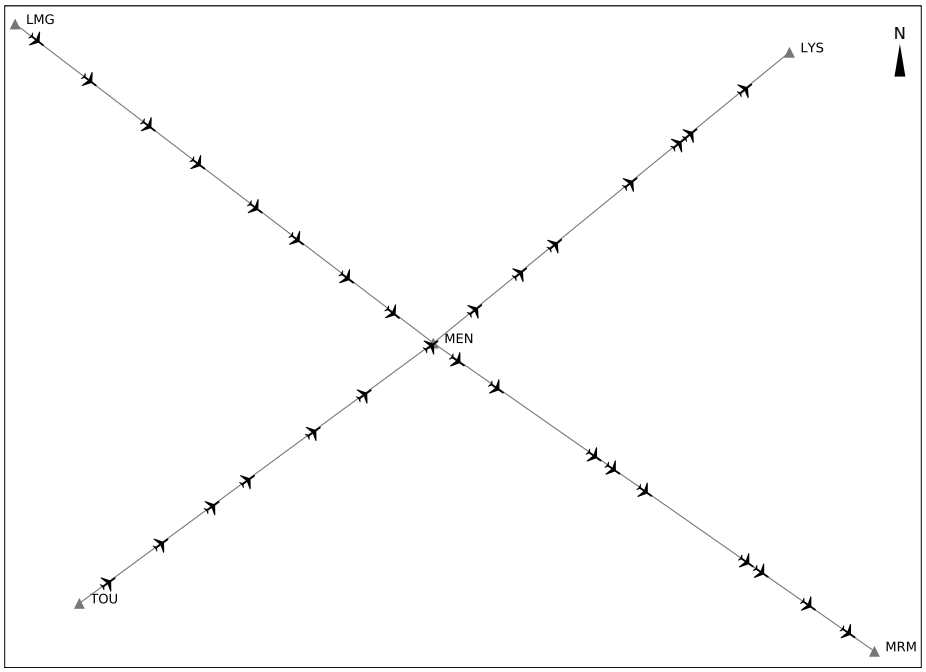

Figure 1: Route network used to simulate the intersection of two flows.

flows are merged before landing. In the topology described in [8], aircraft arrive from two routes ending at the intermediate waypoints WP1 and WP2. Then they head toward the merging point WP3 where the two flows are merged on a single route.

Aircraft can execute two types of maneuvers between WP1/2 and WP3: change speed, and lengthen trajectory by applying a temporary heading change in order to delay arrival time at WP3. Before reaching WP1 and WP2, aircraft make a pairwise negotiation. The first aircraft that will reach WP3 negotiates maneuvers with the second one. When a solution is found, decisions of the first aircraft cannot be modified any more. Then the second aircraft negotiates with the third one, and the process repeats until every conflict is solved. Decisions are taken such that after WP3, all aircraft fly at the same speed and respect separation norms.

During the negotiation phase, the two aircraft iteratively execute an optimization process that minimize trajectory lengthening, speed variation and arrival delay until the conflict is solved.

Even if the method described in [8] and the algorithm described in this Section share some behaviors, they differ in some points. First, these methods do not solve the same problem: while [8] regulates traffic in terminal maneuvering areas, where aircraft decelerate before landing, our method regulates cruising aircraft. Constraints of the problems are different: in terminal areas aircraft

fly at lower altitudes, thus have a larger range of admissible speeds. This point guarantees that all aircraft will be able to fly at the speed required after WP3. Separation distances are also different and change with the aircraft category: for example a heavy aircraft following a light one has to maintain a lower separation distance than when a light aircraft follows a heavy one. While cruising, separation norm is set to 5 NM horizontally.

Both methods regulate speed by computing earliest and latest times of arrival at the target waypoint. But [8] allows a second type of maneuver (lengthening of the trajectory) that increases degree of freedom for the resolution process. On the other hand, the negotiation process is different in our algorithm, in which a decision can be updated after it has been taken. When two aircraft solve a conflict, a third one appearing in the simulation can force the two first aircraft to update their decisions.

Since in [8] an aircraft needs maximum three maneuvers to solve a conflict (two speed changes at WP1/2 and at WP3, and one heading change), it gives better results in terms of number of maneuvers needed to solve conflicts (see Section 2.4). But When the decision is taken, it cannot be updated. On the contrary in the algorithm of this Section, aircraft speed is constantly updated, allowing the traffic to adapt itself in case of disruption (an aircraft failure, a new aircraft entering in the area, etc.).

2.2 Hypotheses

This first algorithm is an attempt to reproduce the Miles-in-Trail traffic structure at the intersection of two flows. In this simulation, aircraft fly along two crossing routes (Figure 1). They perceive position and speed of their neighbors and regulate their speed to avoid conflicts. The size of the neighborhood is an adjustable parameter of the algorithm.

The route network used in this simulation contains two intersecting paths defined by 5 waypoints of French airspace: the first path is composed of the waypoints LMG, MEN and MRM, and the second one by TOU, MEN and LYS. The distances between waypoints are: 116 NM for the segment LMG-MEN, 119 NM for MEN-MRM, 97 NM for TOU-MEN and 105 NM for MEN-LYS. Aircraft are generated randomly at one of the western positions (LMG or MEN). The arrival rate of aircraft along each route follows a Poisson distribution which is considered to be a valid approximation for air traffic flows [35, III–C].

Each flight has a preferred cruise speed depending on general aircraft performances and preferences of airlines. Airlines can give priority to fuel economy by reducing speed, or to the reduction of flight duration, which 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 (which includes crew costs) over the cost of fuel.

Denote v the current speed of an aircraft and v_{opt} its optimal speed. In order to be inserted into a Miles-in-Trail configuration, aircraft may have to choose a speed v different from v_{opt} within a given speed interval. A lower bound $v_{min} = v_{opt} - 6\%$ allows to insert this aircraft into a flow without dramatically

increasing fuel consumption [11]. A speed interval of $[v_{opt} - 6\%, v_{opt} + 3\%]$ is a common choice of projects relying on speed regulation [2].

Optimal speed of generated aircraft are randomly chosen in the interval [447 kt, 487 kt] (kt: knot, stands for NM/s), which are the optimal speeds of an Airbus A320 [1] and an A380 [13]. These aircraft are representative of the ones that are able to fly at 36,000 ft and above, keeping in mind that the A320 belongs to the slowest aircraft of this category, and that the A380, to the fastest (alongside the Boeing 777, for instance). This speed range also guarantees the maximum speed of the slowest aircraft is greater than the minimum speed of the fastest one. Thus, all aircraft simulated in the multi-agent system described in this Section can fly together, which is required to find a solution: for any pair of aircraft, if the fastest aircraft flies at its minimum speed and follows the slowest one that flies at its maximum speed, the first one does not overcome the second one.

Aircraft acceleration and deceleration are fixed to $\pm 4{,}000\,\mathrm{NM/h^2}$ ($0.572\,\mathrm{m/s^2}$) for all aircraft. The standard turn rate [17, PCG S–6] of 3°/s is used so that a complete 360° turn is done in 2 minutes.

Aircraft are able to communicate their position and speed using Automatic Dependent Surveillance-Broadcast (ADS-B In and Out) services and therefore they can know all other aircraft position.

2.3 Algorithm

In this multi-agent system, aircraft are agents exchanging ADS-B messages. They evolve in an environment composed of a route network, represented as a graph whose nodes are waypoints identified by a unique name. In addition to position and speed, in order to save some computation time, ADS-B messages also contain the current edge the aircraft is following in the graph (stored as name of previous and next waypoints). This could be easily deduced from information about geographical position contained in real ADS-B messages, assuming that aircraft follow accurately the route network.

This multi-agent system (Figure 2) is timed by a global clock. Each tick corresponds to a 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 sequentially, agents work logically in parallel. This choice helps to avoid problems related to sequence order. For performance reasons, agents are run in parallel, using multiple threads.

The decision process of each agent is a sequence of three steps. Perception step allows agents to receive ADS-B messages and refresh internal representation of airspace. During decision step, aircraft update their speed on the basis of this internal representation. In action step, agents update their position using updated speed, and broadcast a message containing the information.

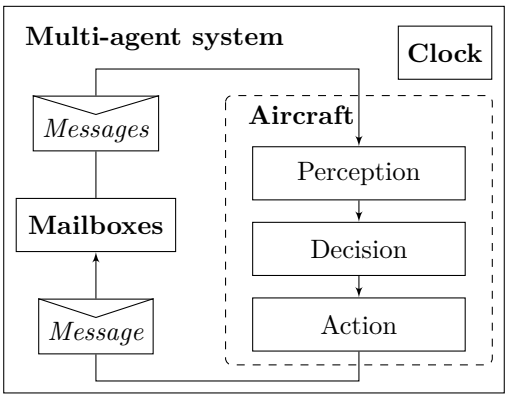

Figure 2: Multi-agent system lifecycle.

2.3.1 Nomenclature

$\boldsymbol{t}$	Current time in the simulation.
$\boldsymbol{P}$	The closest neighbor preceding the aircraft.
$\boldsymbol{F}$	The closest neighbor following the aircraft.
$\boldsymbol{N}$	Any neighbor of the aircraft, including P and F.
$\boldsymbol{t_N}$	Arrival time of the neighbor N to the next waypoint the aircraft will reach.
$\boldsymbol{d_N}$	Distance of the neighbor N toward the next waypoint the aircraft will reach.
$\boldsymbol{v_N}$	Speed of the neighbor N.
$\boldsymbol{v_{\mathrm{opt}}}$	Optimal speed of the aircraft.
$\boldsymbol{v_{\mathrm{min}\mid v_{\mathrm{opt}}}}$, $\boldsymbol{v_{\mathrm{max}\mid v_{\mathrm{opt}}}}$	Minimum and maximum speed allowed by the aircraft performances (i.e. with respect to v_{opt}).
$\boldsymbol{t_{\mathrm{min}\mid v_{\mathrm{opt}}}}$, $\boldsymbol{t_{\mathrm{max}\mid v_{\mathrm{opt}}}}$	Minimum and maximum arrival time to the next waypoint allowed by the aircraft performances (i.e. with respect to v_{opt}).
$\boldsymbol{v_{\mathrm{min}}}$, $\boldsymbol{v_{\mathrm{max}}}$	Minimal and maximal speed found by the decision process of the aircraft described in this Section.
$\boldsymbol{v}$	Speed chosen by the decision process of the aircraft.
$\boldsymbol{S}$	Minimum separation distance between two aircraft.

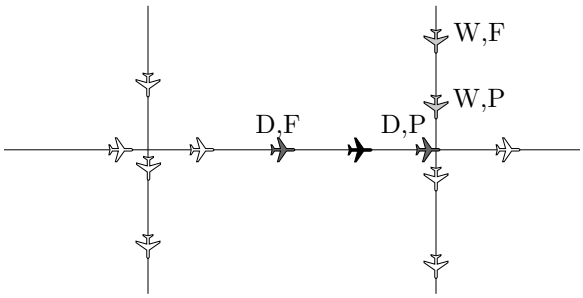

Figure 3: Representation of an aircraft's neighborhood. The black agent receives messages from all surrounding aircraft, detects direct neighbors (D) and waypoints neighbors (W), and whether they are preceding (P) or following (F) it.

2.3.2 Agents' Perception

During the perception step, aircraft first stores information about neighboring agents, using received ADS-B messages (neighbor identifier, position, speed, previous and next waypoints of the flight plan). The aircraft infers the information about the time needed for every neighbor N to reach its next waypoint: $t_N = t + d_N/v_N$, with t the current time, d_N the distance of the neighbor N toward the next waypoint and v_N the neighbor speed.

At a macroscopic level, the Miles-in-Trail scenario described in Figure 1 can be obtained by a 2-steps regulation process. First, both flows of aircraft must be regulated separately by spacing aircraft and by setting identical speeds. Then, they must be synchronized in order to let aircraft pass alternatively the intersection point, while keeping separation minima.

In order to do so, at a microscopic level, each aircraft filters its neighbors to detect 2 pairs (Figure 3): the closest ones (before and after it) in the same flow or more precisely flying from and toward the same waypoints (called direct neighbors in this algorithm) and the pair of aircraft between which it will pass the intersection (called waypoint neighbors).

In order to avoid conflict with direct neighbors, aircraft shall maintain an adequately low speed so as not to overcome its predecessor P, and a sufficiently high speed so as not to be overcome by its follower F.

Waypoint neighbors are the closest ones coming from different waypoints and heading towards the same one as the aircraft, in running order. Aircraft shall fly at an adequately low speed to avoid conflict with its predecessor, and a sufficiently high speed to avoid conflict with its follower.

Computing admissible speed can be seen as finding the intersection of 3 speed intervals:

1. Speed interval physically flyable ($[-6\%, +3\%]$ of optimal speed),
2. Speed allowed by its direct neighbors to avoid conflicts,

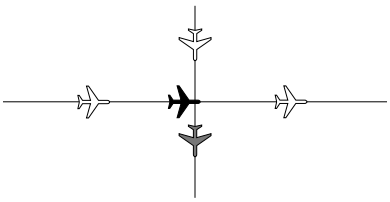

Figure 4: Substitution of missing neighbors. The black aircraft lacks both direct and waypoint predecessors. They are replaced by the last agent that passed its target waypoint (gray).

3. Speed allowed by its waypoints neighbors.

When the aircraft is the first one or the last one on a route section, it may lack of a predecessor or of a follower. The missing neighbor is replaced by the closest aircraft on the other side of this waypoint (Figure 4).

2.3.3 Decision Process

In order to avoid conflicts, two aircraft must maintain a separation distance S. In the case of direct neighbors, an aircraft must be 5 NM behind the next waypoint when its predecessor passes it and 5 NM ahead of this waypoint when its successor passes it ($S = 5\,\text{NM}$). Each of those two pairs of aircraft give an interval of dates where passing the intersection do not generate conflicts.

At a time t, an aircraft is at the distance d of its next waypoint, and wants to compute its maximum speed $v_{\max}$. First, it computes its maximum physical speed $v_{\max|v_{\text{opt}}} = v_{\text{opt}} + 3\%$ (maximum speed with respect to the optimal speed). Maintaining this speed, it will be at a distance $d = S$ before the waypoint at the time $t_{\min|v_{\text{opt}}}$ (minimum time with respect to the optimal speed):

$$t_{\min|v_{\text{opt}}} = t + \frac{d - S}{v_{\max|v_{\text{opt}}}}\,. \tag{1}$$

Then, the aircraft takes into account its predecessor P, such as to reach the same position $d = S$ when $d_P = 0$:

$$t_P = t + \frac{d_P}{v_P}\,, \tag{2}$$

where t_P is the arrival time of the aircraft P at the next waypoint, d_P the distance between P and this waypoint and v_P the speed of the aircraft P.

Combining (1) and (2), at this step, the earliest time of arrival is

$$t_{\min} = \max(t_{\min|v_{\text{opt}}}, t_P)\,. \tag{3}$$

Then, the maximum speed is

$$v_{\max} = \frac{d - S}{t_{\min} - t}\,. \tag{4}$$

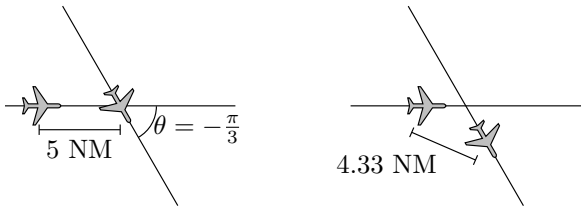

Figure 5: Using a separation distance of 5 NM for waypoint neighbors will lead to a conflict.

Following the same method, the minimum speed $v_{\min}$ is computed, using first $v_{\min|v_{\text{opt}}} = v_{\text{opt}} - 6\%$ (minimum speed with respect to the optimal speed), then substituting the predecessor P by the follower F, $d_A - S$ by $d_A + S$ and the max function by the min:

$$v_{\min} = \frac{d+S}{\min\left(t + \frac{d+S}{v_{\min}}, t_F\right) - t}\,. \tag{5}$$

Once this speed interval $[v_{\min}, v_{\max}]$ has been obtained, aircraft speed can be updated, chosen as close as possible of the optimal speed v_{opt}:

$$v = \begin{cases} v_{\max} & \text{if } v_{\max} < v_{\text{opt}}, \\ v_{\min} & \text{if } v_{\min} > v_{\text{opt}}, \\ v_{\text{opt}} & \text{otherwise.} \end{cases} \tag{6}$$

This method enables an aircraft to take direct neighbors into account. Since all aircraft of a flow apply the same method, every one finally manages to find a speed which avoids conflicts.

In order to avoid conflicts with waypoints neighbors, the same computation is applied a second time, replacing direct neighbors by waypoint neighbors. This second step uses the previously obtained interval $[v_{\min}, v_{\max}]$ (given by the first computation) instead of $[v_{\min|v_{\text{opt}}}, v_{\max|v_{\text{opt}}}]$ (aircraft performances) and arrival dates of waypoint neighbors instead of direct ones.

This two-step computation enables to find the intersection of 3 speed intervals: aircraft performances, speeds enabled by direct neighbors, and speed enabled by waypoint neighbors.

For a pair of direct neighbors (aircraft flying in the same flow), maintaining a separation distance of $S = 5$ NM is enough to guarantee a conflict-free situation. However, for waypoint neighbors, if the same value is used in computation, it will lead to a conflict (Figure 5).

As described in [40, Lemma 1], separation distance S must be increased, depending on relative aircraft headings and speeds:

$$S = \frac{S_0\sqrt{\alpha^2 - 2\alpha\cos\theta + 1}}{\sin\theta}\,. \tag{7}$$

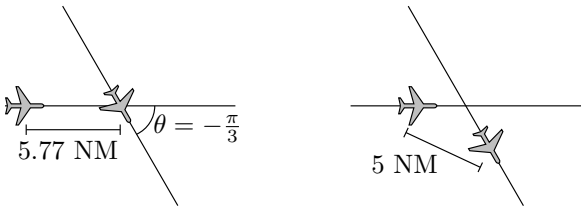

Figure 6: To avoid conflicts between waypoint neighbors, separation distance must be increased.

where $S_0 = 5\,\text{NM}$ is the required separation distance, θ is the angle between both flows and $v = \alpha v_N$. Applied to the situation depicted Figure 6, if both aircraft have the same speed and both fly along trajectories separated by $\theta = \frac{\pi}{3}$, they have to be separated along their trajectory by $S \approx 5.77\,\text{NM}$ in order to be separated by 5 NM at their closest position (Figure 6).

2.4 Results

In order to validate experimentally this algorithm, 2 sets of 3 scenarios were executed, each scenario was run 10 times in order to average results. Aircraft were generated and flying during 1 hour, according to a Poisson point process on each route. Each set of scenarios corresponds to a different average time interval between the generation of two aircraft (parameter λ). Two different parameters λ were tested: 140 s and 110 s. This latter value is the minimum theoretical interval, considering an average optimal speed of 467 kt and a minimal separation distance between aircraft on the same route of 14.14 NM (minimum distance for 2 perpendicular routes). For each λ, three scenarios were tested. In the first scenario, the regulation process was enabled, all aircraft cooperating to solve conflicts. In the second one, the regulation process was disabled, conflicts were not solved. In the third one, 10 % of aircraft were not cooperating, their decision process being disabled. The last scenario allows to validate resilience of the algorithm, by simulating situations where some aircraft cannot or do not want to cooperate, for example because of a failure. Aircraft optimal speeds were chosen randomly within [447 kt, 497 kt].

During simulations, 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 record stores the smallest measured distance between aircraft during the conflict.

Results of each scenario are shown Table 1, values in the table are the average of the 10 runs. Without regulation, minimal distances are spread all along [0 NM, 5 NM], with an average value of 2.3 NM. With regulation, all conflicts could not be solved (only two thirds of them), but the separation distances were almost always greater than 4 NM (except once by run), and the average separation during conflicts was 4.5 NM. This result indicates that this algorithm is able to solve a majority of the initial conflicts, but the resulting separation dis-

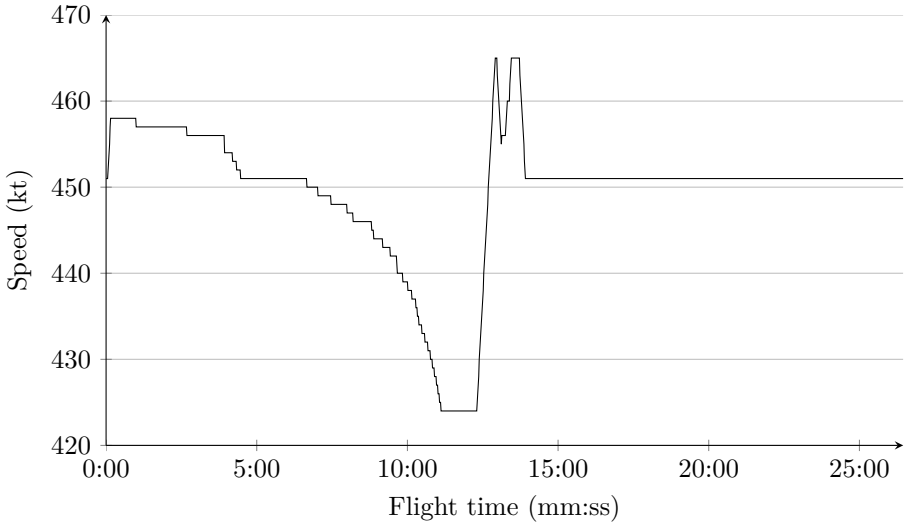

Figure 7: Speed profile of an aircraft in the first algorithm. This aircraft progressively decelerates to avoid a conflict with a waypoint neighbor, then it goes back to its optimal speed during the second half of its flight (after the intersection).

tance is often too close to the minimum of 5 NM. This distance varies constantly around this threshold value: a conflict is counted every time distance becomes slightly lower than 5 NM. Therefore, when the minimum distance measured during a conflict is greater than 4 NM, it can be considered as a false positive.

Thus, if we consider only conflicts where the minimal separation distance is lower than 4 NM, in the first set of scenarios ($\lambda = 140$ s), the algorithm solves 93 % of the conflicts (1 conflict remains of the 13.5 initial ones). In the second set of scenarios ($\lambda = 110$ s), 95 % of the conflicts are solved (1.1 remaining over the 21.7 initial ones). These values indicate that when all aircraft cooperate to the conflict resolution, performances of this algorithm remain stable and close to the optimum until the maximum capacity of this route network is reached (one aircraft every 110 s on average).

On another hand, aircraft speed is adjusted every second to adapt to surrounding traffic. This leads to a high number of small accelerations. In this traffic scenario, the average flight time is 28 min (1,680 s). This means that aircraft spend up to one third of their time accelerating or decelerating (Figure 7). The average difference between current and optimal speeds is smaller in the first set of scenarios (1.2 %) than in the second one (1.4 %). This indicates that a lower rate of aircraft means less constraints for each one: they have more freedom and are closer to their optimal speed (the same remark applies for the number of accelerations).

As described in Section 1.2, resilience is the capacity of a system to recover while confronted to disruptive events. A correctly designed multi-agent system

should be able to recover from such events. The multi-agent system described in this Section aims at solving conflicts. In a nominal situation where all aircraft agents fully cooperate to solve conflicts, this system is able to solve up to 95 % of the conflicts. In a non-nominal situation where a part of the system does not work normally, if the system is resilient it should continue to solve conflicts with a similar efficiency, even if the result quality is slightly lower than in the nominal mode. So resilience of this multi-agent system can be measured as the difference between the number of conflicts when the system works normally and when a part of the system does not work. In order to simulate a degraded mode, we chose to modify the decision process of a part of the aircraft. Failure of these agents is simulated by disabling their decision process. So they are still able to broadcast information about the trajectory they plan to follow, but they do not modify it any more in order to solve conflicts.

The scenarios where the decision process is disabled for 10 % of aircraft shows the resilience of this algorithm to disruptive events, when some agents do not cooperate to avoid conflicts. For the first set of scenarios ($\lambda = 140\,\mathrm{s}$), the number and the severity of conflicts are very similar to the regulated scenarios, where all aircraft cooperate: if we consider only conflicts where separation distance is lower than 4 NM, the average number of conflicts changes from 1 to 1.3 (+2 % while comparing to the 13.5 conflicts of the non-regulated scenario). This value indicates that the algorithm is resilient to local disruptions, like 10 % of non-cooperative aircraft, if the maximum capacity of the route network is not reached. Cooperative aircraft take the non-cooperative ones as a constraint for their decision process, and are still able to solve conflicts by themselves.

In the second set of scenarios ($\lambda = 110\,\mathrm{s}$), the number of conflicts increases by 11 % (from 1.1 to 3.4 over the 21.7 initial conflicts). In this scenario, the maximum capacity of the route network is reached and the algorithm becomes more sensitive to disruptions. Traffic is so dense that each non-cooperative aircraft has a strong impact on the algorithm performances. Nonetheless the algorithm is able to solve 84 % of the conflicts. As long as cooperative aircraft know the position of their non-cooperative neighbors, they will be able to avoid conflicts, because non-cooperative aircraft are taken into account as a constraint in the decision process of cooperative aircraft.

2.5 Limits

This algorithm has several limits. Aircraft constantly update speed to adapt to surrounding traffic. It would be more efficient to plan only a few accelerations: this would lead to reduced speed oscillation and a better predictability of aircraft behavior. In addition, changing the regime of aircraft engines too often reduces their lifetime.

Decisions are based on arrival time at the next waypoint. This is well adapted to route networks composed of long edges (on the order of tens of nautical miles). Still, it works less efficiently for more complex networks with short edges: aircraft may not have enough time while traveling on an edge to adjust speed in order to avoid conflicts. Since they do not consider traffic situation

further than their next waypoint, they are unable to anticipate decisions in dense networks. So this algorithm works for networks used in Miles-in-Trail but not for general route networks.

More generally, Miles-in-Trail have some drawbacks: aircraft are rerouted in order to be put on a network, causing a noticeable lengthening of trajectories. In addition, aircraft in the same flow must fly at the same speed, which can be different from their optimal speed. Both result in an increase of fuel consumption, which can lead to economic consequences for airliners.

Nowadays, aircraft are able to communicate their estimated future 4D trajectory, resulting in a much more accurate trajectory prediction than a simple flight plan based on estimated time of arrival over waypoints. Traffic regulation algorithms should take advantage of this information to compute more efficient solutions.

So as to go beyond those limits, a more generic way to represent the problem of speed regulation for conflict avoidance was found, based on estimation and exchange by messages of full 4D trajectories.

3 Aircraft Speed Self-Regulation in General Traffic

As exposed in [28], many conflict resolution algorithms have already been developed the past decades. 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 [37].

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 [37], instead of trying to reproduce what controllers are doing, those algorithms should help them in an unnoticeable, "subliminal" way. Slightly changing aircraft speed allows automated systems to organize traffic in order to reduce the number of conflicts, creating a "lucky", or favorable traffic situation, more easily managed by controllers. This concept was validated with technical experiments and human factors studies [4, 7].

3.1 Hypotheses

From an algorithmic point of view, this problem can be seen as a generalization of the algorithm described in Section 2, where aircraft adapt speed to avoid (or minimize) conflicts but without the high structuring level imposed by Miles-in-Trail route network: they can follow any 4D trajectory, either following a route network or flying along their optimal trajectory (Free Flight).

Aircraft are ruled by the same constraints as in Section 2 (speed interval of [-6 %, +3 %] of optimal speed, acceleration of $\pm$4,000 NM/h^2, turn rate of 3°/s).

The only modification to these hypotheses is the ability to exchange full estimated 4D trajectories over the next minutes by means of messages. These messages can be transmitted by using Automatic Dependent Surveillance – Contract (ADS-C) [26, 2.2.6]. This protocol is similar to the previously mentioned ADS-B, but instead of requiring aircraft to broadcast position information every fraction of second, those 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 additional information, such as the predicted route composed of a set of 4D positions (3D + time).

Even if current technology does not allow aircraft to really exchange data about trajectories directly from aircraft to aircraft, these data can be collected by ground equipment. The algorithm described below can then compute speed changes which can then be sent back to real aircraft. The distributed aspect of a multi-agent approach to the traffic management is somewhat diminished since aircraft do not take decisions themselves. Yet, other advantages are kept like resilience and parallel computing.

A last difference between the algorithms described in Section 2 and 3 is caused by the problem needing to be solved. A Miles-in-Trail configuration structurally guarantees that a conflict-free solution can be found by regulating speed as long as aircraft are separated enough on their routes. Still, 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 control: the 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 the current implementation of the algorithm, these maneuvers are not implemented: the algorithm only minimizes the number of conflicts and does not try to solve every one of them.

3.2 Algorithm

A trajectory is stored and exchanged as a sequence of arcs that can be segments (aircraft flying at a constant speed, accelerating, or decelerating) and arcs of circles (aircraft turning), as shown in Figure 8. This curve is differentiable at least once everywhere.

This curve is built from a flight plan defined by a set of waypoints. An aircraft flies straight from the first waypoint to the next. It then heads towards the third one, before going straight, and so on. Speed changes are planed at given dates (Section 3.2.2), and are applied in straight sections.

The multi-agent system lifecycle remains the same, as shown in Figure 2. The only modifications are the information contained in messages and the decision process. Therefore, each agent's lifecycle is composed of 3 phases: the first is perception, by which aircraft detects potential conflicts with neighbors; the second is decision: aircraft computes speed changes to avoid conflicts; the

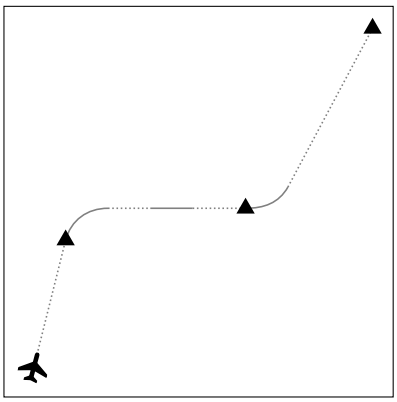

Figure 8: In the algorithm described in Section 3, a trajectory is a curve composed of segments (constant speed as dashed line, acceleration as plain line) and of arcs of circles. Aircraft flies from a waypoint to another (triangles).

third and last phase is action: aircraft updates its position and broadcasts the refreshed 4D trajectory to the neighborhood.

3.2.1 Perception

As in Section 2, 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 (Figure 9).

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

Since an aircraft strictly follows a path, the decision process only modifies speed. To manipulate a trajectory defined by a 4D curve is not necessary: computing the integral of the instant speed as a function of time gives the traveled distance as a function of 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 (Figure 10).

Conflicts detected in the previous step 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 (the intersections of the black aircraft's trajectory and the circles in Figure 9 become the vertical segments in Figure 10).

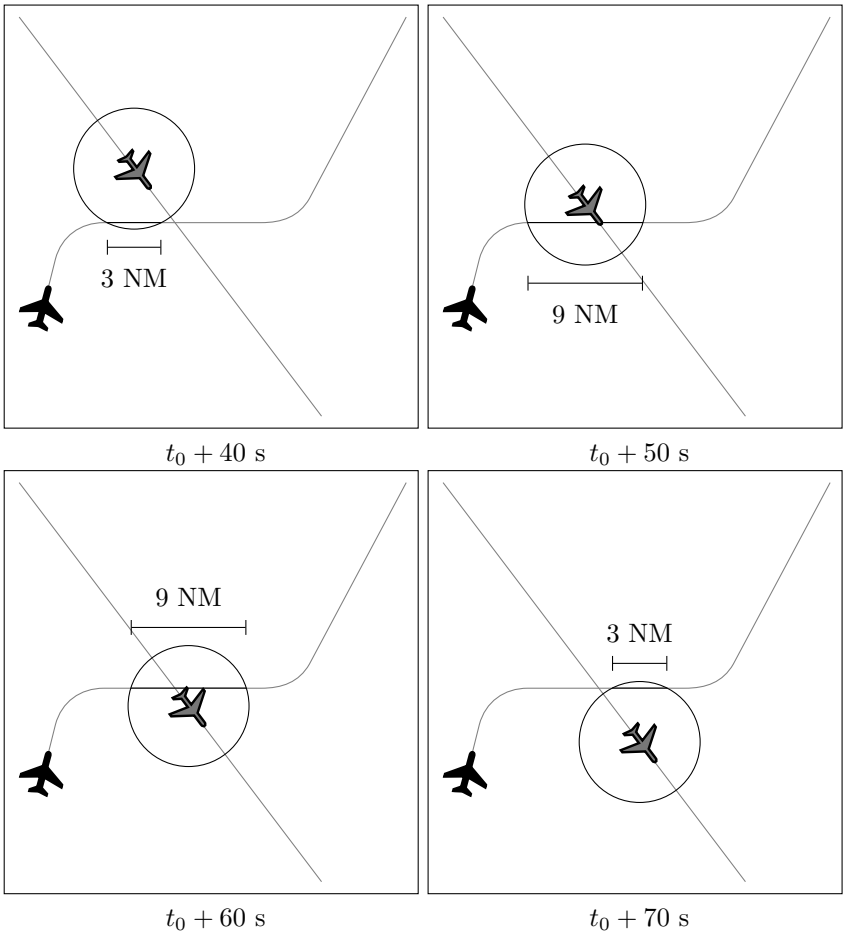

Figure 9: Conflict: intersection between separation distance circle of the gray aircraft and path of the black aircraft.

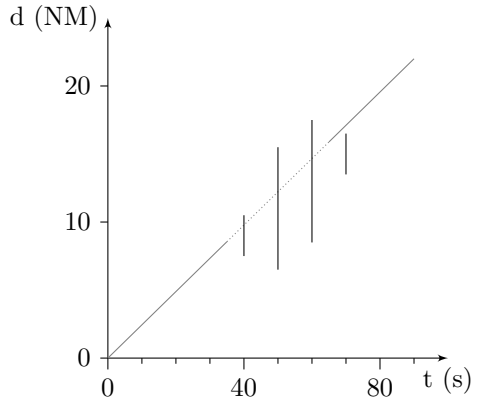

Figure 10: Representation of the 2D trajectory of the black aircraft (distance as a function of time) and intrusions of the gray neighbor in the black aircraft's path at each time sample (vertical segments). A conflict will occur if the black aircraft maintains its initial speed: the curve representing the black aircraft's trajectory intersects with the vertical segments that represents potential conflicts.

3.2.2 Decision

Using conflicts projected in the 2D representation (displayed in Figure 10), aircraft can plan speed changes. The goal is to avoid conflicts by way of speed changes. This goal can be achieved by using a decision tree: for each time sample, the aircraft can maintain its speed, accelerate or decelerate.

To solve this problem, a greedy algorithm was used to plan speed changes. For each time sample, three choices are tested: cruise, acceleration or deceleration at the maximum rate (Figure 11). A time step of 5 s was chosen. Then, the first loss of separation is computed, considering that no more speed changes need to be made afterwards. The choice leading to the trajectory with the latest first loss of separation is validated and the process starts again for the next time step.

As in the following pseudo-code, the set of decisions D is iteratively constructed. For each time step, acceleration (acc), deceleration (dec) and cruise (cr) choices are tested: the first loss of separation is stored in cflTime. Then, the algorithm looks for the decision leading to the latest conflict time. If speed constraints are respected (checked by the function isValid()), this decision is accepted. The resulting set of decisions is then used to compute the new 4D trajectory.

```
1: choices ← {cr, acc, dec}
2: D ← {}
3: for i ← t_0 to t_end step Δt do
4:     bestChoice ← cr
5:     bestCflTime ← i
```

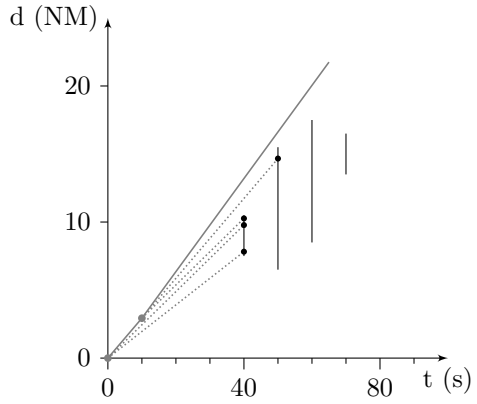

Figure 11: Exploration of the decision tree. For each time step, 3 choices are tested: to accelerate, to cruise, to decelerate. The choice leading to the trajectory with the latest first loss of separation is applied (in this case, it accelerates two times at the beginning, then it cruises).

```
    for c in choices do
        cflTime ← getFirstConflict(D ∪ {c})
        if isValid(D ∪ {c}) & cflTime > bestCflTime then
            bestChoice ← c
            bestCflTime ← cflTime
        end if
    end for
    D ← D ∪ {bestChoice}
end for
```

Using a greedy algorithm makes possible to get good results (Section 3.3) after a short period of computation. Yet, since greedy algorithm is only a local optimization process, it may find a local optimum and be unable to find an existing conflict-free solution. Therefore, some conflicts cannot be solved. The choice of this method is a compromise between the time of computation and the quality of the results.

Results given by the greedy algorithm can nonetheless be refined by the addition of intermediate choices (e.g. accelerations and decelerations of 2,000 $\mathrm{NM/h^2}$ and $\pm$4,000 $\mathrm{NM/h^2}$), but at the cost of longer computation times.

3.3 Results

In order to compare the algorithms described Sections 2 and 3, the same protocol was used again, on the same topology of routes (intersection of two routes, as shown in Figure 1). Aircraft were generated and flying during 1 hour, following a Poisson process on each route. Two different sets of scenarios corresponding to two average time interval λ were tested: 110 s and 140 s. Three scenarios were

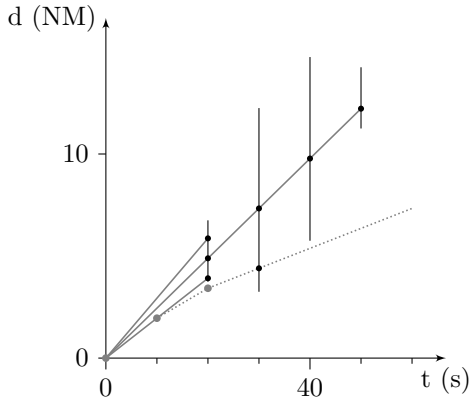

Figure 12: The conflict is unsolvable because of the objective function that tries to maximize the delay before the first loss of separation; in the first time sample, all the possible choices (to accelerate, to cruise or to decelerate) lead to a conflict at $t = 20\,\mathrm{s}$. In this case, the aircraft chooses to cruise, which leads to a severe conflict lasting 40 s. 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).

performed for each time interval: the decision process was first disabled, before being enabled, then disabled for 10 % of the aircraft. This last scenario was added to validate the resilience of this algorithm. Each scenario was composed of 10 runs. The preferred speed of each aircraft was randomly chosen within the interval [447 kt, 497 kt]. The average values of the 10 runs are shown in Table 2.

In the two scenarios where traffic is regulated by the algorithm, for $\lambda = 110\,\mathrm{s}$ and 140 s, the number of conflicts is reduced compared to the results of the method described Section 2, whose results can be seen in Table 1, by $-78\,\%$ for $\lambda = 140\,\mathrm{s}$ (from an average value of 10.3 conflicts to 2.3) and $-63\,\%$ for $\lambda = 110\,\mathrm{s}$ (from an average value of 18.7 conflicts to 6.9). The total time when aircraft accelerate also decreases by $-97\,\%$ (from 410 s to 11.5 s) and $-94\,\%$ (from 503 s to 27.8 s). This indicates that decision process is more efficient when accelerations neighboring aircraft are anticipated.

However, due to the chosen objective in the decision process which is maximizing the time before the first loss of separation, when a conflict cannot be solved, aircraft do not try to minimize its importance (see Figure 12). This leads to lower separation distances (average values of 2.28 NM and 2.86 NM), and a greater number of conflicts where the minimum separation distance is lower than 4 NM. This situation could be improved by introducing conflict duration or predicted minimum distance between aircraft into the objective function or replacing the current criterion by one of them.

Another side effect of the relatively simple decision process we have designed is that aircraft solve conflicts and do not try to optimize speed to get closer

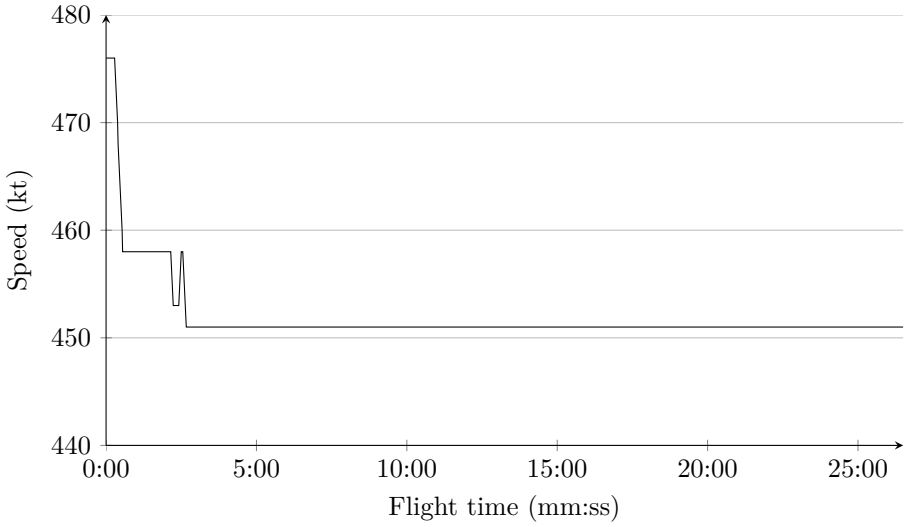

Figure 13: Aircraft speed profile in the second algorithm. This aircraft adjusts speed each time a new neighbor appears before finding a conflict-free speed.

to the optimal one, causing greater distances with the optimal speed. This distance with optimal speed is shown in Figure 13: the aircraft begins to fly at its preferred speed (476 kt), changes its speed several times, then flies at 451 kt until the end, which is far from its preferences. Confronted to the same type of situation, the algorithm described in Section 2 tries to return to its preferred speed, as can be observed in Figure 7. The average distance to optimal speed is then larger in the algorithm described in Section 3 (1.4 % for $\lambda = 140$ s and 2.3 % for $\lambda = 110$ s) than values that the algorithm described in Section 2 returns (1.2 % and 1.4 % respectively). Speed optimization can also be included into the objective function but would lead to a multi-objective problem, harder to model and to solve.

Another set of scenarios was run to validate the second algorithm on actual flight plans of aircraft flying over France during 10 hours (from 4 AM to 2 PM), as shown in Figure 14. Here again, the scenario was tested with conflict resolution first disabled, then enabled. Only the flights at 37,000 ft have been analyzed since this altitude contains the most flights, which are 465 flights in this case. Since the route network is more complex and more dense over France than in the Miles-in-Trail network shown in Figure 1, conflicts can be harder to solve. In some situations, aircraft can start from positions that are too close from each other. This causes unsolvable conflicts (see Figure 15). Nevertheless, this algorithm is able to solve 45 % of the 444 conflicts initially present, leaving in the mean 256.5 unsolved conflicts.

As exposed in Section 1.2, resilience is the ability of a system to recover from disruptive events. For the multi-agent system described in this Section,

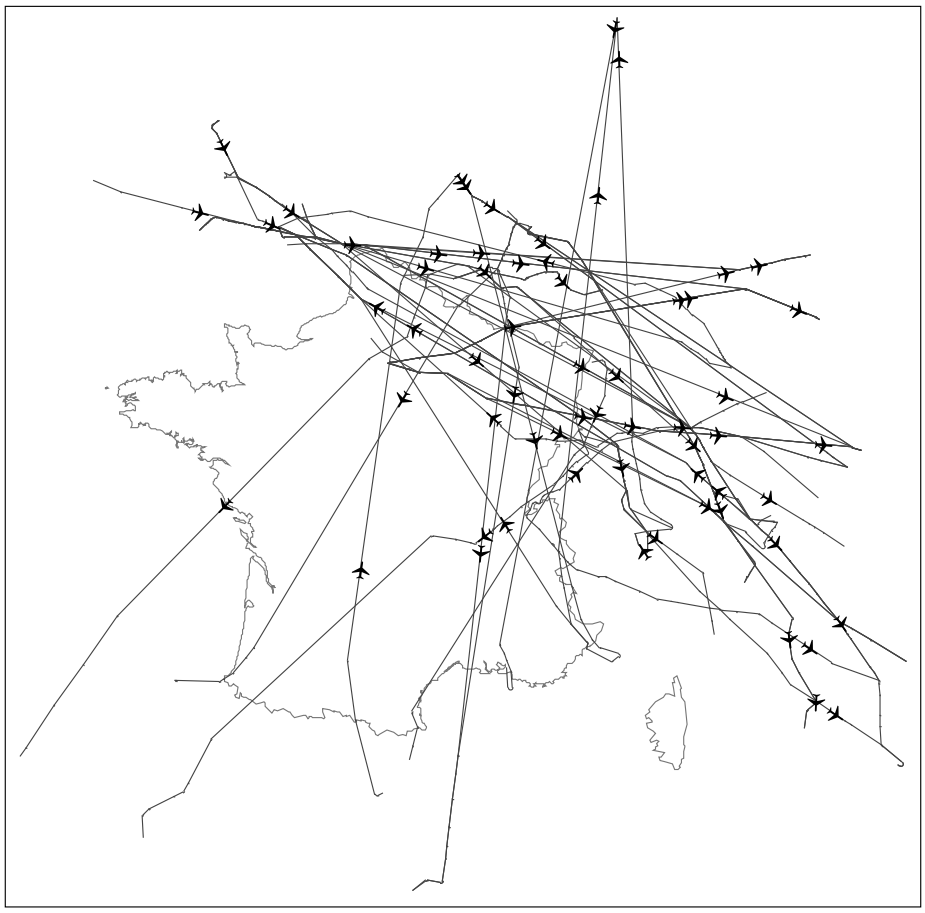

Figure 14: Traffic scenario using actual flight plans, described in Section 3. This picture displays flight plans of aircraft flying at 12:00.

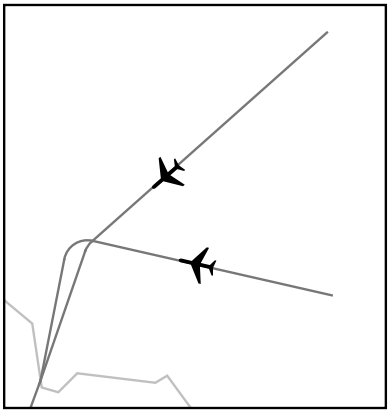

Figure 15: In the scenario based on real flight plans, aircraft are sometimes starting at too close positions and no speed regulation can solve resulting conflict.

resilience is the ability of the algorithm to solve conflicts while a part of the agents are not able to contribute to the collaborative conflict resolution. In order to measure resilience of this system, the validation method described in Section 2.4 was applied to the algorithm of this Section. The decision process of a part of the agents was disabled to simulate a failure: these aircraft are still able to communicate their estimated trajectory, but do not modify it and do not cooperate to solve conflicts. Then, the number of remaining conflicts is measured for the nominal scenario where all aircraft cooperate and the degraded scenario where some aircraft do not cooperate. The difference between these two values indicates the level of resilience of this system.

So as to validate the resilience of the algorithm to disruptive events, each set of scenarios include a scenario where 10 % of aircraft are non-cooperative, their decision process being disabled. For the first random scenario involving two flows of aircraft randomly generated, with an average interval of generations of 140 s, results are similar when every aircraft is cooperative and when 10 % of them are non-cooperative, with an average number of unsolved conflicts increasing from 2.3 to 2.7 (+2 % of conflicts while comparing to the 16.8 conflicts when the decision process is disabled). For the second scenario, with average interval of generations of 110 s, the average number of conflicts increases from 6.9 to 10.7. In this second scenario, aircraft are generated every 110 s, which is the maximum theoretical capacity of the route network. Therefore, each non-cooperative aircraft increases the number of conflicts. For the scenario involving real flight plans, the average number of remaining conflicts also slightly increases from 256.5 to 275.9 (+4 % while these values are compared to the 444 conflicts occurring in the non-regulated scenario). All these values indicate that this multi-agent system is resilient to disruptions, since the cooperative agents take the non-cooperative ones into account in order to include them as constraints

into the decision process.

3.4 Limits

The implementation of ERASMUS concept on subliminal speed changes described in this section has several advantages related to the usage of multi-agent systems. It is resilient to perturbations, like non-cooperative agents, and could eventually be implemented on board, removing the need to rely on ground equipment.

But ERASMUS itself is not able to regulate traffic globally and to solve every conflict (and is not meant to do so). Speed regulation alone cannot solve face-to-face conflicts. The small interval of admissible speeds can be insufficient to maintain separation in all circumstances. To avoid all conflicts, aircraft may need to modify their path.

This is also critical in Free Flight zones, where aircraft can follow any desired path and create even more complex traffic patterns. Thus, another functionality needs to be implemented into this multi-agent system in order to allow aircraft to adjust their path to overcome this situation.

4 Local Minimization of Traffic Complexity

The Free Flight concept aims at giving more freedom to aircraft by allowing them to choose their preferred path without having to fly along the segments of a route network. Free Flight can contribute to decrease airspace congestion, since aircraft can use the whole airspace. Free Flight can also reduce fuel consumption, since aircraft can follow a path minimizing flight time.

The drawback of Free Flight, however, is a possible increase of traffic complexity: since trajectories are not organized on a route network, air traffic controllers experience a higher cognitive workload to understand and regulate traffic [22, 3.1.1]. This is why the implementation of Free Flight is only considered in low-traffic conditions (upper altitudes, night).

In order to facilitate control in Free Flight zones, workload reduction can be done by temporarily creating local route networks in specific areas and letting aircraft follow custom trajectories outside such zones. The algorithm described in this section has two tasks: on one hand, to monitor the traffic complexity in order to detect areas where traffic causes an increase of controllers' workload and on the other hand, to temporarily create route networks in these areas to locally decrease air traffic complexity.

The algorithm was built upon the work described in Section 3. A new agent was inserted into the multi-agent system, as shown in Figure 16. Firstly, this agent collects messages sent by aircraft and containing 4D trajectories. Then, it computes a complexity map using these trajectories to detect areas where complexity is higher than what is allowed. Finally, the algorithm merges trajectories in each detected area in order to create a new local route network. It then sends back path modifications to aircraft that requires it.

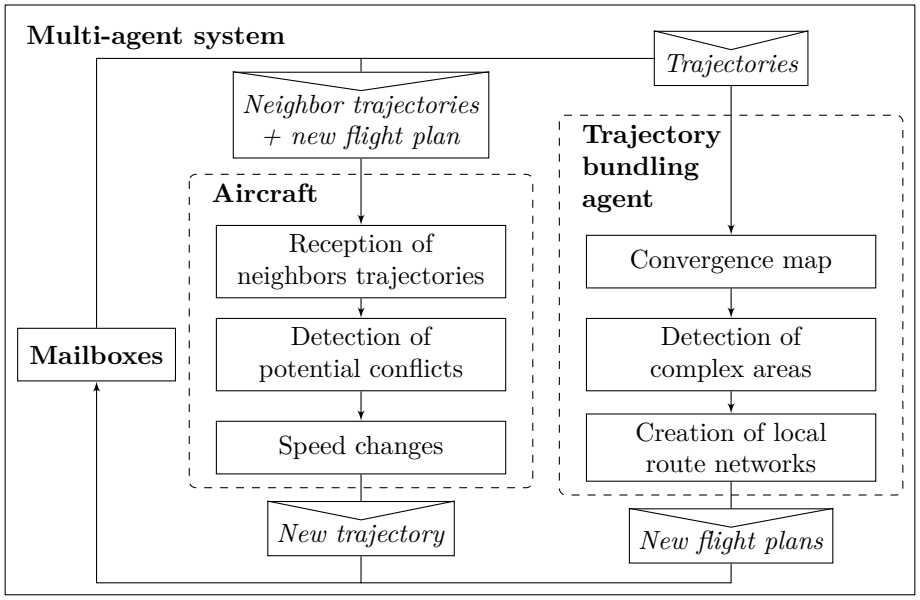

Figure 16: Workflow of the algorithm described in Section 4. A new agent is added to the multi-agent system described Section 3. This agent collects trajectories sent by aircraft using ADS-C messages. It then monitors traffic complexity, before detecting high complexity areas and creating local route networks that reduces complexity. Finally, the new paths are sent back to aircraft, which update their trajectory.

4.1 Detection of Complex Areas

Nomenclature

$\boldsymbol{t}$, $\boldsymbol{t'}$	Times in the simulation.
$\boldsymbol{i}$, $\boldsymbol{j}$	Aircraft.
$\boldsymbol{p_i(t)}$	Position vector of the aircraft i at the time t.
$\boldsymbol{v_i(t)}$	Speed vector of the aircraft i at the time t.
$\boldsymbol{a_i(t)}$	Acceleration vector of the aircraft i at the time t.
$\boldsymbol{p_{ij}(t)}$	Relative position of the aircraft i and j at the time t.
$\boldsymbol{v_{ij}(t)}$	Relative speed of the aircraft i and j at the time t.
$\boldsymbol{p_{ij}(t,t')}$	Relative position of the aircraft i at the time t and j at the time t', used in the robust version of the convergence metrics.
$\boldsymbol{v_{ij}(t,t')}$	Relative speed of the aircraft i at the time t and j at the time t', used in the robust version of the convergence metrics.
$\boldsymbol{r_{ij}(t)}$	Derivative over time of the distance between $p_i(t)$ and $p_j(t)$.
$\boldsymbol{D_{\max}}$	Convergence is computed between aircraft separated by less than this maximum distance.
$\boldsymbol{\Delta t}$	Robust convergence uses only positions within a time interval $[t - \Delta t, t + \Delta t]$.
$\boldsymbol{C_{ij}(t)}$	Convergence between $p_i(t)$ and $p_j(t)$.
$\boldsymbol{C_{ij}}(t, t')$	Convergence between $p_i(t)$ and $p_j(t')$, used in the robust version of the convergence metrics.
$\boldsymbol{\widetilde{C_{ij}}(t)}$	Robust convergence computed between the positions $p_i(t)$ and $p_j(t')$, where $t' \in [t - \Delta t, t + \Delta t]$.
$\boldsymbol{\widetilde{C_i}(t)}$	Robust convergence computed at the position $p_i(t)$, considering all aircraft $j \neq i$.
$\boldsymbol{C}$	Convergence map, as a 2D grid.
$\boldsymbol{(k,l)}$, $\boldsymbol{(m,n)}$	Coordinates in the grid C.
$\boldsymbol{C_{k,l}}$, $\boldsymbol{C_{m,n}}$	Convergence value at the coordinates (k, l) (resp. (m, n)) in the map C.

Complexity is measured by using convergence metric developed by Delahaye and Puechmorel (2000) [10, Section 3]. The usual way to monitor the complexity of a traffic situation is the congestion metric that measures the traffic density. However, this indicator and controllers workload can be uncorrelated. The convergence metric is an attempt to give a more accurate complexity indicator, compared to the congestion metric. The main idea is the following: if two aircraft converge to the same position, a conflict may occur, and therefore this situation may increase the controller's workload. Otherwise, if aircraft diverge, no action is required and the workload does not increase.

Let $p_i(t)$ be the position of the aircraft i at the time t, and $v_i(t)$ its speed vector. The relative position $p_{ij}(t)$ and speed $v_{ij}(t)$ of aircraft i and j at the time t are given by

$$p_{ij}(t) = p_j(t) - p_i(t)\,, \tag{8}$$

$$v_{ij}(t) = v_j(t) - v_i(t)\,. \tag{9}$$

For each time t, the derivative of the distance between two aircraft i and j is given by

$$\begin{aligned} r_{ij}(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\|p_{ij}(t)\| \\ &= \frac{p_{ij}(t)\cdot v_{ij}(t)}{\|p_{ij}(t)\|} \\ &= \|v_{ij}(t)\|\cos(p_{ij}(t), v_{ij}(t))\,. \end{aligned} \tag{10}$$

Equation (10) is homogeneous of degree zero with respect to p_{ij}. Since computing convergence of aircraft separated by hundreds of miles is meaningless, convergence only takes into account vectors within a spatial window of size $D_{\max}$. r_{ij} is positive when aircraft diverge, and it is negative when they converge. The divergence does not contribute to the traffic complexity, so the convergence $C_{ij}(t)$ is the following:

$$C_{ij}(t) = \begin{cases} -r_{ij}(t) & \text{if } r_{ij}(t) < 0 \text{ and } \|p_{ij}(t)\| < D_{\max}, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Estimation of future aircraft position is subject to a strong uncertainty [39], and depends on unpredictable factors such as wind. The error on lateral position can be neglected and the aircraft is supposed to be able to accurately follow its path. However, the error on longitudinal position is influenced by speed fluctuations following a normal distribution with a standard deviation of 15 kt, or 15 NM after 1 h. A robust evaluation of the complexity has to take this longitudinal uncertainty into account. The average of convergences is computed within a temporal window of size $\Delta t = \frac{15\,\mathrm{NM}}{420\ \mathrm{kt}} \approx 128\,\mathrm{s}$, considering that 420 kt is the minimum aircraft speed in this simulation.

Equations 8, 9, 10 and 11 must be modified to take into account this time uncertainty. We add a second argument to these functions which is a time t' belonging to $[t - \Delta t, t + \Delta t]$:

$$p_{ij}(t, t') = p_j(t') - p_i(t)\,, \tag{12}$$

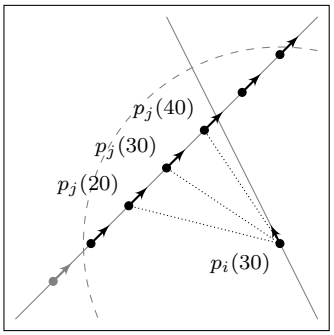

Figure 17: Computation of the convergence value $\widetilde{C_{ij}}(30)$, located at $p_i(30)$. A first filter is applied to keep $p_j(t)$ within the spatial window $D_{\max}$ represented by the dashed circle, and the time interval Δt of $\pm 10\,\mathrm{s}$. Convergence is computed separately between $p_i(30)$ and the remaining $p_j(t)$ (i.e. $p_j(20)$, $p_j(30)$ and $p_j(40)$), then averaged.

$$v_{ij}(t,t') = v_j(t') - v_i(t)\,, \tag{13}$$

$$\begin{aligned} r_{ij}(t,t') &= \frac{p_{ij}(t,t') \cdot v_{ij}(t,t')}{\|p_{ij}(t,t')\|} \\ &= \|v_{ij}(t,t')\| \cos(p_{ij}(t,t'), v_{ij}(t,t'))\,, \end{aligned} \tag{14}$$

$$C_{ij}(t,t') = \begin{cases} -r_{ij}(t,t') & \text{if } r_{ij}(t,t') < 0 \text{ and } \|p_{ij}(t,t')\| < D_{\max}, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Then, in order to compute robust convergence values between aircraft i and j at the time t, convergence $\widetilde{C_{ij}}(t)$ is computed as the average convergence of the vectors $v_i(t)$ with all the vectors $v_j(t')$, where t' belongs to $[t - \Delta t, t + \Delta t]$:

$$\widetilde{C_{ij}}(t) = \text{average}(\{C_{ij}(t,t') | t' \in [t - \Delta t, t + \Delta t]\})\,. \tag{16}$$

Averaging values within a time window allows to reduce the uncertainty related to the time dimension of aircraft trajectories. An illustration of this step is shown in Figure 17. Finally, convergence $\widetilde{C_i}(t)$ is computed as the sum of convergences between the aircraft i and all its neighbors j at the time t:

$$\widetilde{C_i}(t) = \sum_{i \neq j} \widetilde{C_{ij}}(t)\,. \tag{17}$$

In this way, convergence generated by three converging aircraft will be the double of convergence generated by two aircraft.

$\widetilde{C_i}(t)$ is an isolated convergence value at the position $p_i(t)$. In order to compute the convergence map C, all these isolated values must be transformed into a 2D grid in which each cell corresponds to a spatially located area. The value at coordinates (k, l) is computed as the maximum value among the values

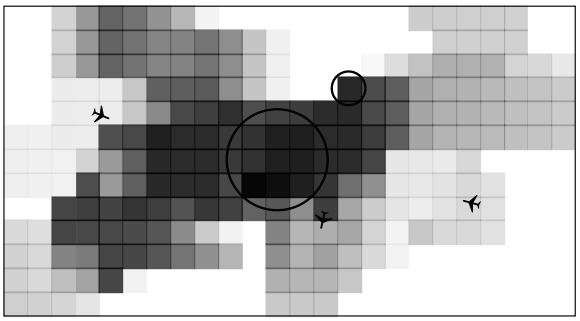

Figure 18: Convergence map (lower values in white, higher ones in black) with detected areas needing to be regulated (circles).

$\widetilde{C_i}(t)$ located in the cells of coordinates (m, n) surrounding the cell (k, l):

$$C_{k,l} = \max(\{\widetilde{C_i}(t) | p_i(t) \in (m,n), \\ m \in \{k-1, k, k+1\}, \\ n \in \{l-1, l, l+1\}\}) \tag{18}$$

This algorithm was integrated into the multi-agent system described in Section 3. Even if we were not able to decentralize completely this algorithm, the first computation steps can be distributed among aircraft agents. An aircraft i knows trajectories of its neighbors, received by ADS-C messages like in Section 3. It can compute convergence values $\widetilde{C_i}(t)$ along its trajectory and use them to compute partial maps. These partial maps are sent in a message to the traffic monitoring agent, as shown in Fig. 16. The monitoring agent aggregates these maps, keeping the maximum value at each coordinates to obtain the final convergence map.

Finally, in order to determine areas needing to be regulated, the convergence monitoring agent detects cells whose value is greater than a given threshold. It then forms clusters of contiguous cells and draws the smallest surrounding circle (see Figure 18). Towards the end of this step, intersecting circles are merged. Temporary route networks are then created in those circles.

4.2 Design of a Local Route Network in High Complexity Areas

Nomenclature

$\boldsymbol{t}$	A time value. For each trajectory, t is normalized to belong to $[0, 1]$.
$\boldsymbol{i}$	Aircraft identifier.
$\boldsymbol{N}$	Number of aircraft.
$\boldsymbol{p_i(t)}$	Position vector of the aircraft i at the time t.

$\boldsymbol{v_i(t)}$ Speed vector of the aircraft i at the time t.

$\boldsymbol{a_i(t)}$ Acceleration vector of the aircraft i at the time t.

$\boldsymbol{l_i}$ Length of the trajectory i.

$\boldsymbol{\Omega}$ Space in which density is computed.

$\boldsymbol{x}$ Position in the space Ω.

$\boldsymbol{d(x)}$ Density value at the position x, computed using a kernel density estimator.

$\boldsymbol{H(\Omega)}$ Entropy computed for the probability density function $d(x)$.

$\boldsymbol{K_h}$ Scaled kernel of bandwidth h used to compute the probability density function.

$\boldsymbol{\eta_i(t)}$ Move to apply to one position $p_i(t)$ in order to minimize entropy of the set of trajectories.

Structuring traffic on a route network is a way to reduce complexity at the cost of flight efficiency. To achieve this, bundling methods can be used to merge trajectories into flows. In this case, bundling is used to create a temporary route structure from a set of aircraft trajectories in order to oblige aircraft to follow a route network. Several bundling methods have been developed [20, 29] for data visualization purpose. They bundle similar trajectories into a network of major aircraft flows to visualize them, but does not guarantee that bundled trajectories are geometrically correct (flyable) without excessive curvature. Puechmorel and Nicol [31, 32] developed a method to bundle trajectories by minimizing entropy of curves. This method produces geometrically correct (flyable) trajectories.

In information theory, entropy measures complexity of a system. If an event has a high probability of occurrence, occurrence of this event does not add much information to the system. In other words, a system in which some events occur with a high probability is less complex than a system with many equiprobable events. Applied to aircraft trajectories, if all aircraft use the same route, the system is less complex than if all trajectories differ.

A probability density can be computed from a set of aircraft trajectories. Entropy can be computed from this probability density. If aircraft share portions of their trajectories because they follow a route network, entropy value is lower than if trajectories are spread evenly in the airspace. Reciprocally, minimizing entropy of the probability density of a set of trajectory will bundle trajectories into a route network. Puechmorel and Nicol [31] used a gradient descent method to minimize this entropy by deforming trajectories in order to bundle them.

The trajectory of an aircraft i can be represented as a curve defined by a function $p_i(t)$ that returns aircraft position from arc length. The time samples t are normalized to belong to the interval $[0, 1]$. Speed $v_i(t)$ and acceleration $a_i(t)$ are then the first and second derivative of $p_i(t)$.

Entropy of a probability distribution measures its concentration and is maximized on unbounded interval by uniform distributions. Shannon entropy, from information theory, is applied to discrete probability distributions. By analogy, entropy H can also be computed for a continuous probability density $d(x)$, where x is a point of a space Ω:

$$H(\Omega) = -\int_{\Omega} d(x)\log(d(x))\,\mathrm{d}x\,. \tag{19}$$

An example of entropy computed for a discrete probability distribution is shown in Table 3.

Thus, one can measure entropy based on the density of traffic: if trajectories are structured on a route network, the density is locally concentrated and the entropy based on traffic density is low. Otherwise, if traffic is unstructured and all aircraft choose a different route, entropy will be high. Therefore, measuring and minimizing entropy based on density enables to structure traffic in order to bundle trajectories.

In this case, to measure traffic density, a kernel density estimator is used to get a continuous probability density. This method estimates a continuous density as the average of a set of kernel functions centered on the data sample. In our case aircraft positions $p_i(t)$ are extracted along the trajectories, then a 2D kernel is centered on each position. An Epanechnikov kernel $K_h(x)$ was chosen. For N trajectories, the density $d(x)$ is given by the following function:

$$d(x) = \frac{1}{N}\sum_{i=1}^{N}\int_0^1 K_h(\|x - p_i(t)\|)\,\mathrm{d}t \tag{20}$$

$$K_h(u) = \frac{2}{\pi h^2}\left(1 - \left(\frac{u}{h}\right)^2\right)\mathbf{1}_{\{|u|<h\}}\,, \tag{21}$$

where h is the bandwidth of the kernel, which defines its size. The parameter h must be chosen large enough such that the resulting probability density is smooth, the representation of each trajectory being continuous in the resulting probability density, but not too large such that individual trajectories are still distinct. The kernel bandwidth also affect the number of resulting flows. A large bandwidth (several tens of nautical miles) tends to bundle all the trajectories in a single flow. A narrow bandwidth (a few miles or less) tend to create several flows distant from each others, since kernels from different trajectories do not intersect. In our implementation we chose to sample trajectories every 40 s and to use a bandwidth of 5 NM. An example of density computed from a set of trajectories is shown in Figure 19.

Yet, computing a density map this way gives too much importance to portions of trajectories while aircraft are slower, as when they take off and land. Thus, density must take velocity into account:

$$d(x) = \frac{1}{\sum_{i=1}^{N} l_i}\sum_{i=1}^{N}\int_0^1 K_h(\|x - p_i(t)\|)\|v_i(t)\|\,\mathrm{d}t\,, \tag{22}$$

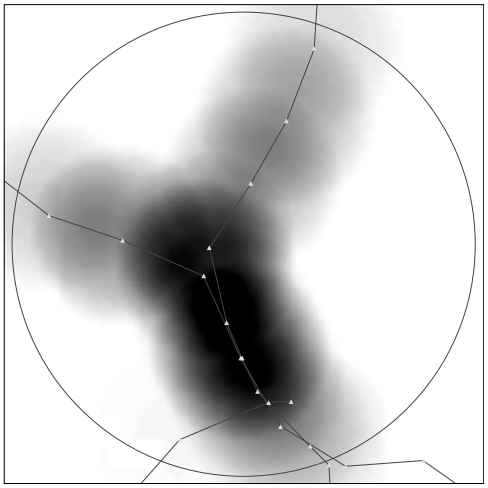

Figure 19: Density computed from a set of trajectories, used in the entropy minimization process. Trajectories are in black. Lighter areas represent low density, darker ones represent high density.

where l_i is the length of trajectory p_i.

To bundle those trajectories, entropy must be minimized by using a gradient descent method. The first order variation of the entropy given by (19) defines the movement $\eta_i(t)$ to apply to each point $p_i(t)$ of trajectories [31], $()_{\mathcal{N}}$ being the component of the vector normal to the trajectory and $K'_h(u)$ the derivative of the kernel function:

$$\eta_i(t) = \int_\Omega \left(\frac{p_i(t) - x}{\|p_i(t) - x\|} \right)_{\mathcal{N}} K'_h(\|p_i(t) - x\|) \log d(x) \, \mathrm{d}x \|v_i(t)\| \tag{23a}$$

$$- \left(\int_\Omega K_h(\|p_i(t) - x\|) \log d(x) \, \mathrm{d}x \right) \left(\frac{a_i(t)}{\|v_i(t)\|} \right)_{\mathcal{N}} \tag{23b}$$

$$+ \left(\int_\Omega d(x) \log d(x) \, \mathrm{d}x \right) \left(\frac{a_i(t)}{\|v_i(t)\|} \right)_{\mathcal{N}} . \tag{23c}$$

A detailed explanation of the behavior of (23) can be found in [32]. The general result of (23) is a minimization of the entropy. The line (23a) results in a local maximization of the density. The lines (23b) and (23c) tend to minimize trajectory curvature by moving each point $p_i(t)$ along the normal vector, resulting in the minimization of the total trajectory length.

To implement this method, the convergence monitoring agent first samples trajectories inside each regulated circular area that is detected during the previous step of the algorithm, described in Section 4.1 (see Figure 18). The samples are extracted every 40 s. Then, the agent computes the density map, using a bandwidth of 5 NM for the kernel function, in such a way that trajectories representation are continuous in the density map (kernels centered on successive

samples overlap). Each sample, except the first and last one of every trajectory, is moved according to (23). A coefficient is applied to reduce movements and avoid divergence caused by too large displacements. The computations for each sample can be done in parallel, since it relies only on density map.

The two last steps (computing density map and moving points) are done iteratively until convergence is observed (maximum movement of a waypoint during an iteration is lower than a given threshold) or when a maximum number of iterations is reached.

The resulting sample positions define the waypoints of the portion of the temporary flight plans that are sent back to aircraft.

4.3 Results

A test scenario was built to validate the method. Three parallel routes of 100 NM and spaced by 5 NM are crossed by another one, causing convergence. The intersection of those trajectories is detected as an area to regulate (circle in Figure 20). Trajectories are then bundled in this area by the entropy minimization algorithm. Entropy of the resulting trajectories is lower inside the circular area than in the initial situation according to (23). Density is maximized because all the trajectories are bundled. Curvature is at the same time minimized: trajectories are straightened and no excessive curvature appear.

New flight plans are sent back to aircraft, which follow them. As shown in Figure 16, the conflict resolution algorithm detailed in Section 3 works in parallel with the trajectory bundling agent. Aircraft compute and broadcast trajectories avoiding conflicts. The trajectory bundling agent creates temporary route networks in high complexity areas, and sends new flight plans to aircraft. Then aircraft update their trajectories according to these flight plans and update speed changes in order to avoid conflicts. New trajectories are broadcast, and the process is repeated until the end of the simulation.

4.4 Future works

This method gives promising results for elaborating temporary local route networks. Trajectories are correctly aggregated to form simpler paths. Yet, two aspects have to be improved in future work.

Firstly, since entropy measure is based on density, aircraft heading is not taken into account. In this 2D modeling, two trajectories that form an obtuse angle are then aggregated into a single path, leading to a face-to-face conflict. In an operational context, this type of conflict do not appear, two aircraft with an opposite heading fly at different altitudes. An improvement of this algorithm would be to implement this type of behavior.

Secondly, in this implementation, the choice has been made to centralize decision process. A central monitoring agent was introduced into the multi-agent system described in Section 3. The lifecycle of this agent is divided into three steps. The convergence map is computed during the perception step. Detection of zones to regulate and aggregation of trajectories are done during

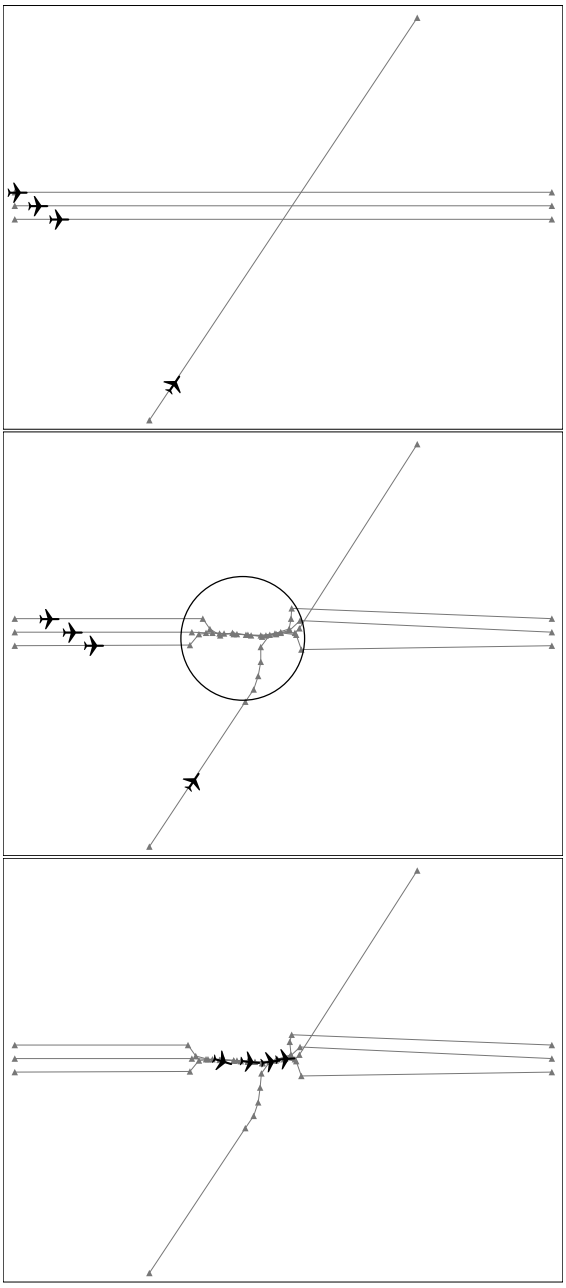

Figure 20: Results of the third algorithm. Top: initial scenario, central trajectories are 100 NM long, and spaced by 5 NM. Center: aggregated flight plans, area to regulate represented by the circle. Bottom: aircraft following new flight plans.

the decision step. New flight plans are sent back to aircraft in messages during action step.

This architecture improves the time of convergence but one of the main assets of multi-agent systems is lost: it relies on a single agent to take all the decisions. This process could be decentralized at aircraft level: convergence can be computed by each aircraft along its trajectory and distributed by messages in order to collaboratively compute the whole convergence map. The same principle can be applied to trajectory modification process but will need further investigations.

5 Conclusion

In this article, three decentralized methods enabling to organize some aspects of air traffic were presented. The algorithms described in Section 2 aims at solving conflicts on a highly structured route network (Miles-in-Trail). The algorithm described in Section 3 aims at reducing the number of conflicts in a general traffic configuration. The algorithm described in Section 4 creates temporary route networks where traffic complexity increases locally. While being promising, these methods can be improved to fulfill operational requirements.

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.

References

[1] Air Canada. Our fleet: Airbus a320-200 (320). `http://www.aircanada.com/en/about/fleet/a320-200xm.html`. Accessed: 2015-08-25.

[2] 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.

[3] 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*, pages 225–242, 2008.

[4] 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.

[5] J. M. Canino, J. García, J. M. Molina, and J. B. Portas. *A Multi-Agent Approach for Designing Next Generation of Air Traffic Systems*. INTECH Open Access Publisher, 2012.

[6] D. Capera, M.-P. Gleizes, and P. Glize. Self-organizing agents for mechanical design. In *Engineering Self-Organising Systems*, pages 169–185. Springer, 2004.

[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*, 18(6):963–974, 2010.

[8] R. Chipalkatty, P. Twu, A. R. Rahmani, and M. Egerstedt. Merging and Spacing of Heterogeneous Aircraft in Support of NextGen. *Journal of Guidance, Control, and Dynamics*, 35(5):1637–1646, Sep 2012.

[9] 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, July 1996.

[10] D. Delahaye and S. Puechmorel. Air traffic complexity: towards intrinsic metrics. In *Proceedings of the third USA/Europe Air Traffic Management R & D Seminar*, 2000.

[11] 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.

[12] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, Nov 2006.

[13] Emirates. Emirates a380 specifications. `http://www.emirates.com/english/flying/our_fleet/emirates_a380/emirates_a380_specifications.aspx`. Accessed: 2015-08-25.

[14] EUROCONTROL. Free route airspace. `http://www.eurocontrol.int/articles/free-route-airspace`, Oct 2016.

[15] European Commission. Commission Implementing Regulation (EU) No 1207/2011 of 22 november 2011 laying down requirements for the performance and the interoperability of surveillance for the single european sky. Official Jourrnal of the European Union, Nov 2011.

[16] Federal Aviation Administration. 14 CFR Part 91: Automatic Dependent Surveillance – Broadcast (ADS–B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule, May 2010.

[17] Federal Aviation Administration. *Aeronautical Information Manual, Official Guide to Basic Flight Information and ATC Procedures*, apr 2014.

[18] K. M. Feigh, A. R. Pritchett, A. Shah, S. Kalaver, A. Jadhav, D. Holl, R. Bea, and A. Gilgur. Analyzing air traffic management systems using agent-based modeling and simulation. 2005.

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

[20] M. Gariel, A. Srivastava, and E. Feron. Trajectory clustering and an application to airspace monitoring. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1511–1524, Dec 2011.

[21] S. M. Green. En route spacing tool: Efficient conflict-free spacing to flow-restricted airspace. In *ATM2000 3rd USA/Europe Air Traffic Management R&D Seminar, Napolo*, 2000.

[22] B. Hilburn. Cognitive complexity in air traffic control: A literature review. *EEC note*, 4(04), 2004.

[23] J. Hoekstra, R. van Gent, and R. Ruigrok. Designing for Safety: the 'Free Flight' Air Traffic Management concept. *Journal of Reliability Engineering & System Safety*, 75(2):215–232, 2002.

[24] International Civil Aviation Organization. *Procedures for Air Navigation Services. Air Traffic Management*. Number 4444-ATM/501. Fifteenth edition, 2007.

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

[26] International Civil Aviation Organization. *Global Operational Data Link Document (GOLD)*, apr 2013.

[27] D. Isaacson, A. Sadovski, and D. Davis. Tactical scheduling for precision air traffic operations: Past research and current problems. *Journal of Aerospace Information Systems*, 11(4):234–257, 2014.

[28] J. Kuchar and L. Yang. A review of conflict detection and resolution modeling methods. *Intelligent Transportation Systems, IEEE Transactions on*, 1(4):179–189, Dec 2000.

[29] A. Marzuoli, C. Hurter, and É. Féron. Data visualization techniques for airspace flow modeling. In *CIDU 2012, Conference on Intelligent Data Understanding*, pages pp 79 – 86, Boulder, United States, Oct. 2012.

[30] A. Pritchett, S. Lee, S. Verma, and A. Jadhav. Examining air transportation safety issues through agent-based simulation incorporating human performance models. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 2, pages 7A5–1–7A5–13 vol.2, Oct 2002.

[31] S. Puechmorel and F. Nicol. Entropy minimizing curves with application to automated flight path design. June 2015.

[32] S. Puechmorel and F. Nicol. Entropy minimizing curves with application to flight path design and clustering. working paper or preprint, July 2016.

[33] 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*, pages 1–16, May 2013.

[34] O. Rodionova, M. Sbihi, D. Delahaye, and M. Mongeau. North atlantic aircraft trajectory optimization. *Intelligent Transportation Systems, IEEE Transactions on*, 15(5):2202–2212, Oct 2014.

[35] E. Salaün, M. Gariel, A. Vela, and E. Feron. Aircraft Proximity Maps Based on Data-Driven Flow Modeling. *ArXiv e-prints*, Jan. 2011.

[36] D. Sislak, P. Volf, and M. Pechoucek. Agent-based cooperative decentralized airplane-collision avoidance. *Intelligent Transportation Systems, IEEE Transactions on*, 12(1):36–46, March 2011.

[37] J. Villiers. *Automatisation du contrôle de la circulation aérienne: "ERASMUS", une voie conviviale pour franchir le mur de la capacité.* Etudes & documents - I.T.A. ITA, 2004.

[38] 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*, pages 1–11, 2004.

[39] L. C. Yang and J. K. Kuchar. Prototype conflict alerting system for free flight. *Journal of Guidance, Control, and Dynamics*, 20(4):768–773, 1997.

[40] J. Yoo and S. Devasia. Decoupled conflict resolution procedures for non-perpendicular air traffic intersections with different speeds. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 275–280, Dec 2013.

Table 1: Performances of the first algorithm (average values of 10 runs).

	$\lambda = 140$ s			$\lambda = 110$ s		
Decision process	**Disabled**	**Enabled**	**Disabled for 10 % of aircraft**	**Disabled**	**Enabled**	**Disabled for 10 % of aircraft**
Number of aircraft	51.2	51.3	52.2	65	65	66.6
Number of conflicts	16.8	10.3	8.6	27	18.7	20.1
Minimum distance (NM)	0.03	2.99	0.22	0.0	2.28	0.0
Average distance (NM)	2.31	4.47	4.3	2.33	4.49	4.25
Number of conflicts < 4 NM	13.5	1	1.3	21.7	1.1	3.4
Average number of speed changes	0	410	370	0	503	470
Average distance to optimal speed (%)	0	1.2	1.1	0	1.4	1.3

Table 2: Performances of the second algorithm (average values of 10 runs).

	$\lambda = 140$ s			$\lambda = 110$ s			Real traffic		
Decision process	**Disabled**	**Enabled**	**Disabled for 10 % of aircraft**	**Disabled**	**Enabled**	**Disabled for 10 % of aircraft**	**Disabled**	**Enabled**	**Disabled for 10 % of aircraft**
Number of aircraft	51.2	51.1	51.3	65	65	65.8	465	465	465
Number of conflicts	16.8	2.3	2.7	27	6.9	10.7	444	256.5	275.9
Minimum distance (NM)	0.03	0.04	0.01	0.0	0.02	0.0	0.0	0.0	0.0
Average distance (NM)	2.31	2.28	2.08	2.33	2.86	2.5	2.5	2.29	2.36
Number of conflicts < 4 NM	13.5	1.5	2	21.7	4.2	7.4	342	216.9	227.9
Average number of speed changes	0	11.5	11.2	0	27.8	23.2	0	28.6	25.8
Average distance to optimal speed (%)	0	1.4	1.4	0	2.3	2.3	0	2.4	2.2

Table 3: Example of entropy computed from two discrete probability distributions, using Equation (19). Entropy is maximized when all the events are equiprobable.

	x	1	2	3	4	5	**Entropy**
Non-equiprobability	$d(x)$	0.8	0.05	0.05	0.05	0.05	**0.78**
Equiprobability	$d(x)$	0.2	0.2	0.2	0.2	0.2	**1.61**