An Empirically grounded Agent Based simulator for the Air Traffic Management in the SESAR scenario

– Full Model Description –

Introduction

This document describes the model developed by the ELSA project – Empirically grounded agent based models for the future ATM scenario – funded by the Work Package E of SESAR (Single European Sky ATM Research Joint Undertaking). This description is the most up to date of the full model. Its implementation in C and Python can be freely downloaded at [1]. The technical description of the implementation and a short user manual can be found at [2]. Additional information can be found in the deliverables of ELSA, two of which are freely downloadable at [3] and [4]. Finally, an article describing the context and presenting some results has been submitted to Transportation Research Part B: Methodological.

The ELSA Air Traffic Simulator is composed of several fairly independent modules:

- A **network generator**, including navigation points and sectors.
- A **strategic layer**, used to test different traffic situations and generate "planned" traffic,
- A **rectification module**, used to straighten up trajectories when simulating free-routing,
- A **tactical layer**, with a conflict resolution engine, simulating a tunable, imperfect super-controller.
- A **post-processing** module, including standard metrics computation and a simple graphic interface to see isolated run.

A schematic view of the model in presented in figure 1.

I. NETWORK GENERATOR

The ELSA simulator requires a schematic representation of the airspace, which is a superposition of a network of sectors and a network of navigation points. The simulator is able to work both with real networks and artificially generated ones. The latter are introduced in order to have some control on the simulation of the model. More importantly, the user can mix real features and synthetic ones in order to test to airspace configurations or idealized situations.

A. Overview

The simulator is based on 2d representation of the airspace. The generator creates independently two networks on a plane, one for sectors and one for navigation points. Each of these networks can be bases on data given by the user, or generated from scratch. More specifically the user can:

- Specify the boundaries of sectors or let the generator create them (specifying their number and the type of network, see hereafter).
- Specify the capacities of the sectors, or let the generator infer it from traffic data (using maximum number of flights in the area).
- Specify the position of the navigations points, or let the generator create them randomly.
- Specify the possible flying routes between navigations points and the times needed, or let the generator infer them from data, or ask the generator to create them from scratch (based on different networks, see hereafter).

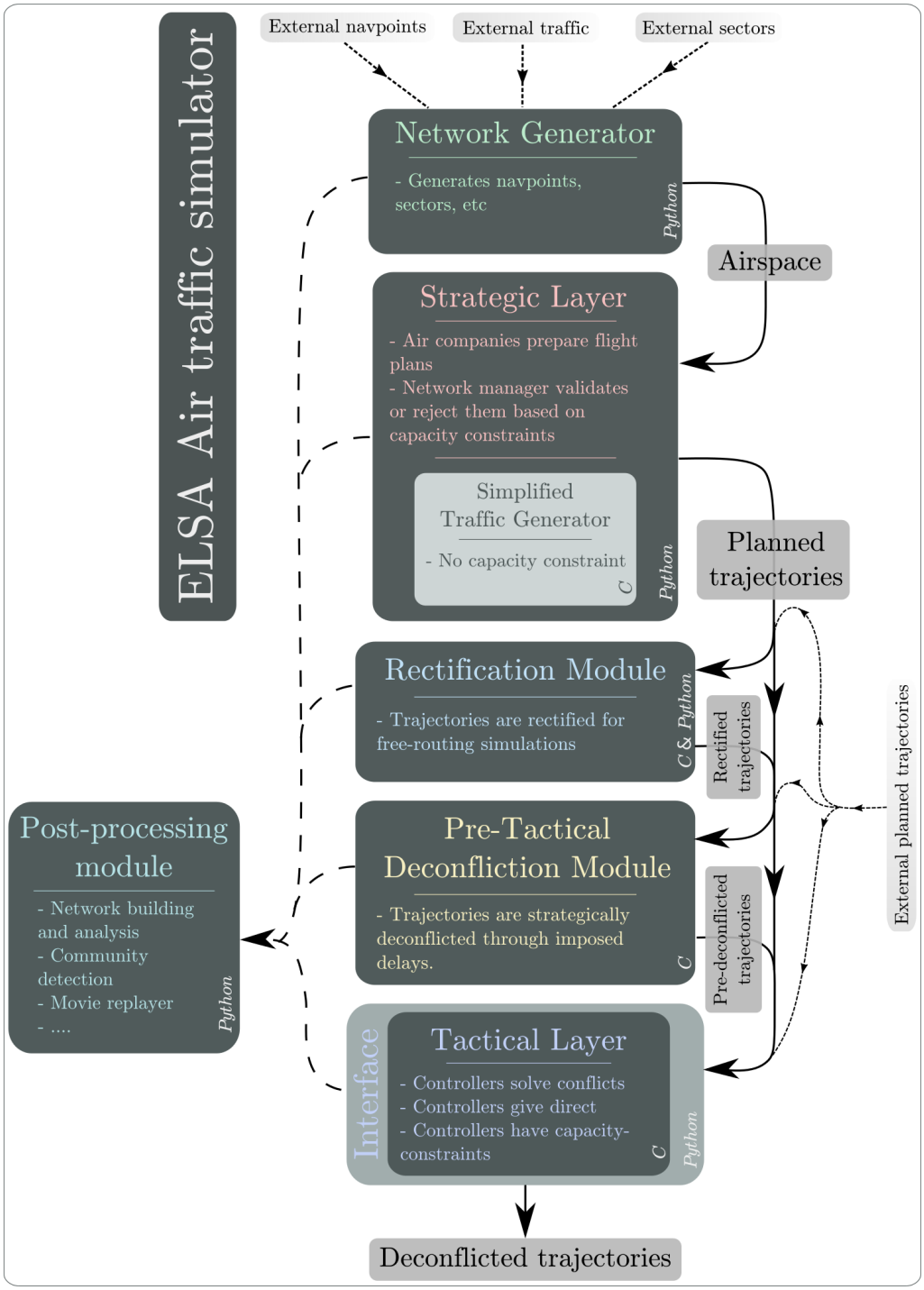

FIG. 1: Organization of the model.

- Specify the entries/exits of the airspace, or let the generator infer it from traffic data, or ask it to choose them randomly.

Hence a whole spectrum of artificial networks are possible, ranging from purely artificial to purely data-driven. One can for instance use real sectors, generate the navpoints randomly, choose the flying times from a given distribution, add manually the entries/exits, or whatever other combinations.

B. Modelling details

Hereafter we present the rationales for the choices we made for the model.

- We identified the navigation points and the sectors (elementary sectors and collapsed sectors) as the main components of the airspace. Thus, the simulator uses two networks. In the first one, each node represents a navigation point and an edge between two nodes indicates the possibility for a flight to go from one to the other. This is done either by using as input the real traffic flow, or by performing a Delaunay triangulation (see I C). In the second network, each node is a sector, and an edge is present if the two areas of the sectors represented by the nodes have a common boundary. Note that each navigation point is uniquely associated with a sector, so that each trajectory of navigation points can be converted to a trajectory on the network of sectors (but not the other way around).

- In our model, time is a continuous variable, thus the crossing times, i.e. the time needed to go from one navigation point to the other, are real numbers. When fed with traffic data, the generator takes as crossing time between two navigation points the average crossing time of all the real flights travelling between the two considered nodes. Note also that our navigation point network is directed and therefore for each pair of connected navigation points we have two crossing times depending on the travel direction. The crossing time between two connected navigation points can also be computed from scratch, based on the Euclidean distance between the two navigation points. A constant of proportionality can then be tuned so that the average crossing time matches some value chosen by the user. Finally, it can be extracted from a Gaussian distribution, with tunable parameters.

- Regarding capacity, the generator uses simple scalar values attached to each node (sector). These values represent the maximum number of flights crossing the airspace in hour one. The user can choose manually the values of the capacities, or leave it to the generator to compute them from traffic data, using the maximum number of aircraft in an hour. Clearly this is an underestimation of the real capacity (if sectors always operates below the maximum capacity), but in practice it gives a good approximation. The generator has an option to base the capacities on other metrics too, for instance the area of the sector.

- The network is static, each node (sector) being defined for the whole duration of the simulation.

- The entry/exit points, as well as the airports, are specially labelled and need to be fixed during the construction of the model. The first way of doing this is to pass a file containing the labels of the navigation points which should be considered as entry/exit. The second way is to let the generator infer the entries/exits from traffic data. Finally, the user can let the generator choose select the entries and exits. It picks at random some navigation points randomly for this.

- Once the entry/exit points are set, the generator computes the possible **pairs** or entries/exits. Since can be done automatically (choosing all possible pairs), with some limitations on the minimum number of navigation points between the entry and the exit. It can also be done manually, based on a file passed to the generator. Finally, it can be inferred from some traffic data.

- The decisions of the agents in the strategic layer are based on the best paths on the network of navigation points. This is why the network generator computes the best paths on the network between all possibles entries and exits. In fact, it even computes the k-best paths on the network for each pairs, k being equal to the number of flights plans submitted by air companies during the strategic layer (see section II A). The detection of the k-best paths is based on Dijkstra's and Yen's algorithms [5, 6].

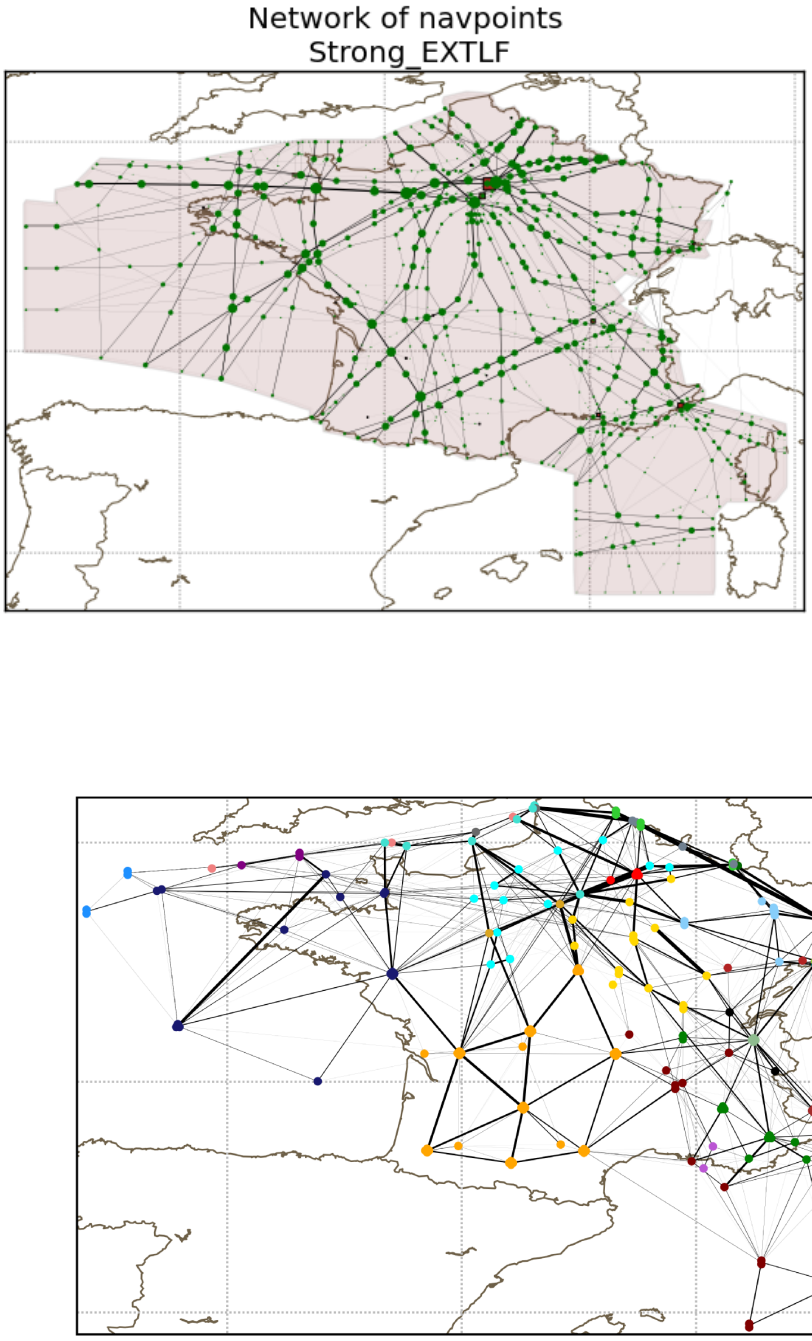

FIG. 2: Traffic network of navigation points (top panel) and sectors (bottom panel) for France: each navigation point/sector is represented by a node. Two nodes are linked if at least one flight goes from one to the other and the link is weighted proportionally to the number of flights. The colors on the bottom panel represent pieces of ACCs the sector belongs to.

C. Artificial Networks

Concerning the purely synthetic networks, three different graph structures are used to emulate the network of sectors and navigation points:

- Delaunay triangulation of a set of randomly chosen points (Fig. 3, left),
- Triangular lattice, which is a regular graph (Fig. 3, center),
- Erdos-Rényi random graph, which is a non planar graph (Fig. 3, right).

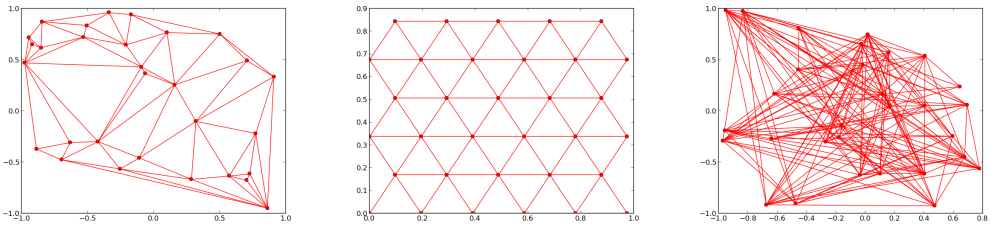

FIG. 3: Left: example of a Delaunay triangulation of 30 random points in a square region of the plane. Center: example of a triangular lattice of 30 points in a square region of the plane. Right: example of an Erdos Rényi random graph of order (*i.e.* number of nodes) 30.

Given N points on a plane, a Delaunay triangulation for them is a collection of edges satisfying an "empty circle property", that is, a subdivision of the plane in triangles such that no element of the point set is inside the circle passing through the three vertices of any triangle (*i.e.* the circumcircle). The minimum of the angles of each triangle, used for the triangulation, results maximized in order to provide the best possible covering of the plane with the minimum number of triangles. This particular triangulation is named after the Soviet/Russian mathematician Boris Delaunay[7].

An alternative way of defining the Delaunay triangulation of a discrete set of points is through its connection with the Voronoi tessellation. This tessellation, named after the Ukrainian/Russian mathematician Georgy Voronoi, is a way of dividing a plane into N regions. In order to build a Voronoi tessellation, one first chooses a set of points (termed "generators") in the plane. Then for each one of them one constructs a surrounding tile as the region consisting of all the points closer to that generator than to any other element in the set.

The Delaunay triangulation and the Voronoi tessellation are in a dual relationship. In fact, starting from a Voronoi tessellation one can build another graph that has a vertex corresponding to each tile. For each tile boundary in the Voronoi diagram one then draws an edge joining two neighbouring tiles. The obtained graph is nothing but a Delaunay triangulation. The term dual is used because this property is symmetric and works also the other way around. As figure 4 shows, the Delaunay triangulation can be easily interpreted in terms of sectors. In fact, each generator point is the centroid of a sector, here defined as one of the tile in the Voronoi tessellation. The triangulation is thus the natural way of building a network of sectors by joining two neighbouring sectors.

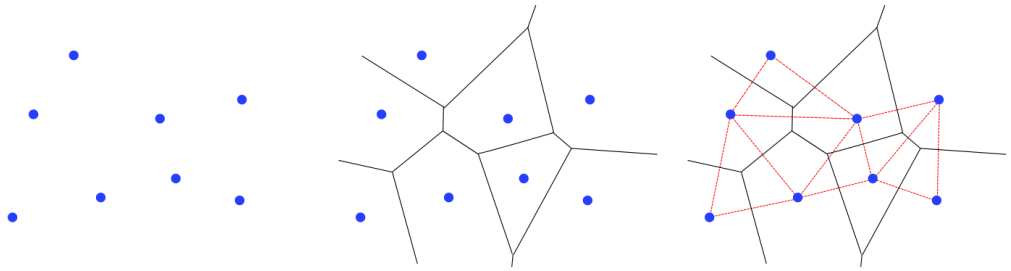

FIG. 4: Voronoi tessellation of 8 randomly distributed points on plane. First, we draw randomly some points on the plane (left). Then we build the Voronoi tessellation (center) by drawing the perpendicular bisectors between each pairs of points, stopping the line when it crosses another one. All points in one given tile are now closer to the corresponding vertex than to any other vertex (definition of the Voronoi cell). Finally, the Delaunay triangulation can be easily drawn (right) by putting a link between every pairs of vertices which have one common boundary. Voronoi and Delaunay networks are dual and uniquely defined. Moreover, the Delaunay triangulation is the best triangulation one can get for a set of points.

In order to build the Delaunay triangulation, the code generates a set of uniformly distributed points in a square region of the plane. Then it performs a Voronoi tessellation. As stated above, in such a tessellation the points inside the boundaries of each tile are nearer to its centroid point than to each other point in the

starting set. Each tile represents an airspace sector. The Delaunay triangulation is obtained by plotting the dual graph associated with the Voronoi tessellation. Finally this Voronoi tessellation is unique, as well as the Delaunay triangulation. The Delaunay triangulation is clearly a planar graph and each vertex has on average six surrounding triangles (*i.e.* the mean degree is equal to 6, which is also the maximum average degree for planar graph).

Starting from a set of points (randomly or not), one can build also a random graph. Such kind of graph is obtained starting from a set of nodes and adding edges between them at random. For this the generator uses the model developed by Erdos and Rényi [8]. In this model the graph is built by drawing an edge between two nodes with a probability p independent from every other edge.

One can fix the mean degree $\langle k \rangle$ of the nodes by fixing $p = \langle k \rangle / (N-1)$. By default, the mean degree is set to 6, i.e. the same one than the one obtained with a Delaunay triangulation. Note that using a random graph as a model for the sector network means to relax the planarity condition. In general, the random graph cannot be drawn in such a way that no edges cross each other.

II. STRATEGIC LAYER

In the strategic layer, we model the interaction process between air companies and the network manager (CFMU) in order to choose the allocation of flights in the airspace, given the constraints due to capacity and/or safety restrictions. More specifically, in this phase of the model, we assume that a specific airspace is given, together with its navigation point and sector structure, airport location, and sector capacities. Then air companies submit to the network manager a number of possible flight plans for each desired flight. The way in which air companies select the flight plans is driven by an optimization process, where a company-specific cost function is used. The network manager then selects which flight plans are viable, given the current constraints and allocations of the airspace. Finally, the air company selects the best among the accepted flight plans. The model is based on an Agent-Based Modelling (ABM) paradigm.

The strategic layer is based on two types of agents, the air operators (AO) and the network manager (NM), that we describe in the following.

A. Agents - Air Operators

The main logical building blocks in the modeling of AOs are:

- The AO performs a simple, local optimization of the flight plans. This means that a company optimizes the trajectory only for one flight at a time. Optimization takes into account the length of the flight and the departure (or arrival) time, but does not consider the other flight plans' characteristics. In particular, we do not take into account the fact that the flight plan needs an available aircraft. Indeed, we are interested in the last part of the optimization of the air companies. Once they have computed their need in aircraft, crew and potential passengers, they choose the actual path of the flights.
- There is no interactions between the strategies, a consequence of the fact that the optimization is local, i.e. done separately for each flight.
- We consider all the aircraft as equivalent in terms of speed and flight characteristics.
- We can have an arbitrary number of AOs, possibly having different utility/cost functions used in the optimization.
- There is no time memory, i.e. past allocations of airspace do not have any role in the decision of the Network Manager or in the choice of the AOs.

The AOs are supposed to perform several tasks in a proactive way. At each time step a company is chosen randomly and it selects randomly a pair of airports and a desired departing time t_0 for the flight (see below for details on the possible distributions). The AO considers the N_{fp} best flight plans (FPs) according to its cost function. Then it submits them in sequential order from the best to the last to the NM until one is accepted. The accepted FP modifies the availability of the airspace for the next AOs contacting the NM.

The process of choosing the N_{fp} best flights plans is a complicated issue where the network of navigation points comes into play into the optimization procedure. The selection of the best flight plans presents some subtleties.

When considering the network of navigation points, one might naively think to compute the N_{fp} shortest paths. These paths would then be converted by the NM in paths in the network of sectors to check their viability, given the

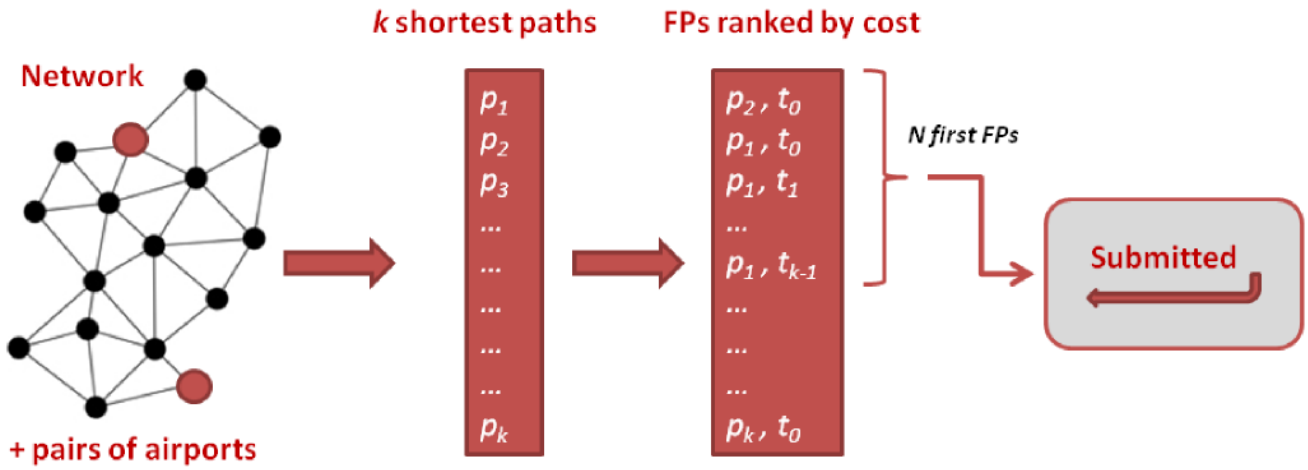

FIG. 5: The figure shows the N_{fp} flight plans selection operated by an AO: given the network and the flight (as a pair of departing/arriving airports), the N_{fp} shortest paths are determined. Different combinations of paths and departing times are tried and their costs are computed. The combinations are sorted following decreasing cost; the N_{fp} with higher rank are selected to represent the array of proposal for the given AO and for the given flight.

current traffic. However, if this is done by using realistic values of the number of navigation points per sectors, one obtains only very few different sector paths, namely one or two for $N_{fp} = 10$. Indeed, since there are typically more than 20 navigation points per sector, it is quite easy to build different navigation points paths (which can differ only by one nodes and have similar length) within the same path of sectors. In terms of traffic regulation this is a problem because if one sector path is not allowed because of a capacity limit, it is likely that another navigation point path with the same network path (even if with slightly different entry and exit times) will be forbidden. Therefore we need to design a AO's strategy of selection of the N_{fp} best flight plans, which allows to explore a larger variety of sector paths.

In order to do so, we introduced another parameter, N_{sp}, which is the maximum number of navigation point shortest paths for each sector path. For instance, if $N_{sp} = 1$, we compute first the N_{fp} shortest paths of sectors, then for each of them, we restrict the network of navigation points to the nodes contained in the sectors[**?**] of that path, and finally we compute the shortest path of navigation points. With this process, we end up in general with $N_{fp} \times N_{sp}$ paths, among which we choose the N_{fp} best ones. Note that even if we take $N_{sp} = N_{fp}$, which in theory gives only one sector path, in reality we might end up with more, because there are not enough navigation point paths taking this sector path. In this case, the algorithm looks for navigation point paths in another sector path.

Let us now move on to how the AO chooses the flight plans among the possible ones. First, the AO chooses at random a pair of airports among the available ones and a desired departing time t_0. Then it computes N_{fp} (spatial) paths to the arriving airport, using the process described above. After that, the AO tries a series of combinations of the previously found paths by shifting them in time. For instance, given one of the shortest paths, it is shifted by one or two time steps of length τ[**?**]. Each path associated with a departing time becomes a "flight plan". After that, the AO sorts the flight plans by computing the values of a predefined cost function $c(p_i, t_0^i)$. Here t_0^i is the departing time of the i^{th} flight plan and p_i is the spatial path, i.e. the array of navigation points. Note that in general t_0^i is different from the desired departing time t_0. Finally, it submits to the NM the N_{fp} flight plans one by one starting with the minimal cost until one of them is accepted, if any (see Fig. 5).

The AO's cost function depends on two variables:

- The weighted length of the path l *i.e.* the sum of the weights of the edges (i.e. crossing times) of the path. This is also exactly the spatial length of the flight, since we assume constant speed.
- The difference between the time of departure t_0^k and the originally desired time of departure t_0.

The selected AO cost function reads:

$$c(p_k, t_0^k) = \alpha l + \beta \left| t_0 - t_0^k \right| . \tag{1}$$

The ratio between α and β characterizes the type of AO. In fact, a company that values most the length of the flight (for example, because of fuel consumption) will have a cost with an high value of α compared to β. For example

a low cost carrier without a hub and spoke structure would probably be described by such a cost function. On the contrary, a company that values most the punctuality will have an high value of β. This is the case of a major carrier with a networked hub and spoke structure of flights. Since the cost function is a linear combination of two terms and due to the fact that what matters for the process is the ranking of the costs of the flight plans, what really matters is only the ratio between α and β. For this reason in most of the following analyses we will set arbitrarily $\alpha = 1$ and we vary β.

Finally, note that for any company the best flight plan is always the one with the shortest path and the no time shifting ($t_0^i = t_0$).

The setup of the desired times is an important component of the model. In fact, the user can set an arbitrary distribution for the desired departing times t_0 during the day. Four cases are implemented in the simulator:

- All flights have the same desired departing time.
- Desired departing times are drawn from an uniform distribution between two given times (e.g. the beginning and the end of the day).
- Desired departing times are drawn from a distribution with a certain number of peaks.
- Desired departing times are taken from the real data.

For the last case, note that in general one does not have direct information on the desired departing time, but only the actual last filed flight plan (M1 file). The departure times in the data *are not* the desired times, but merely the already regulated times. We use the departing time as a proxy of the desired departing time. However since the M1 file are regulated to avoid capacity overload, the simulator can add a noise term to the actual departure time. This noise is drawn from a normal law centred in zero and with a tunable standard deviation δt_0.

The ABM supports an arbitrary number of types of Air Operators, each of them having a different cost function, i.e. different parameters α and β. These parameters are permanently associated with the type of company. In fact, only the ratio β/α of the cost function is important to choose.

Note that there are two extreme types of AO strategies:

- the first one has a high ratio β/α, meaning that for these companies it is very important to be on the desired time. Hence, they prefer to choose an alternative route rather than shifting in time. This type of companies is called "company R" – for "rerouting".
- the second one has a small ratio β/α, resulting in a preference for a shifting in time if the previous flight plan is rejected. These companies is called "company S" .– for "shifting".

Note also that, for any ratio, the best first flight plan is the one with the shortest path and with a time of departure equal to the desired time.

B. Agents - Network Manager

The main logical building blocks in the modelling of the NM are:

- The NM is passive and performs a local optimization. Essentially it answers to the AOs by communicating if the flight plan of a given flight is allowed. In case of rejection, the AO submits a new flight plan which is considered by the NM.
- The M0 phase is modelled through a first round of submission and allocation of the airspace. Then the network can be disturbed (i.e. some sectors can be shut down) and when it happens, the AOs whose flights are affected by the sector closure resubmit new flight plans. This process leads to the M1 trajectories, which are the equivalent of the last filed flights plans.

In our model the NM acts by defining a random queue among the AOs. Their proposed flight plans are considered following the order settled by the queue (random order). The NM reacts to each AO proposal by checking which FPs do not cross full sectors. In practice, this is done by checking one FP after the other beginning with the best one (according to the AO). The first allowed FP is allocated to the airspace, but, of course, it can happen that there is no accepted flight plans. It is important to note that one does not need to assume that the NM knows the ranking of the company, but simply tests all the submitted FPs, communicating to the AO which are allowed, and the company will then choose among them the one with the lowest cost.

In the case where a sector is shut down, the NM finds out which companies are impacted and asks them to resubmit a new set of flight plans on the new network (i.e. the old one without the sectors which are shut down). It then recreates a queue, keeping the priority defined in the original queue, and checks for each flight plan, as before.

C. Metrics

Strictly speaking, the output of the strategic layer is a bunch of trajectories simulating M1 flight plans. However some tools are also included to compute some important metrics, that we describe here.

The first metric is the number of flight plans which are rejected by the network manager. More precisely, we measure the average number of flight plans rejected per flight. In a well performing system this number should be as small as possible. Clearly this quantity gives also a measure of the degree of agreement between AO desires and what the NM is able to give them. In fact the more flight plans are rejected, the less likely it is that the chosen flight plan is good for the AO, since flight plans are sorted by increasing cost.

The second metric is the average of the number of flights for which all flight plans have been rejected. In other words, we consider a flight as non-regulated if at least one of its N_{fp} flight plans has been accepted. This measure is complementary with the previous.

In addition to the two previous metrics, the strategic layer computes the following satisfaction. For each flight f, the satisfaction of an AO is defined as:

$$\mathcal{S}_f = c_f^{best}/c_f^{accepted}, \tag{2}$$

where c_f^{best} is the cost of the optimal flight plan for the flight f according to the AO cost function (this flight is also the first flight plan to be submitted for the flight), and $c_f^{accepted}$ is the cost of the flight plan accepted for this flight. If no flight plans has been accepted, we set S_f to 0. Note that S_f is then always between 0 and 1. The value 1 is obtained when the best FP is accepted.

The satisfaction is a metric defined for a single flight. The strategic layer also compute the average satisfaction across all the flights of a given AO with label i, i.e:

$$\mathcal{S}^(i) = \frac{1}{M^{(i)}} \sum_{i=1}^{M} \mathcal{S}_f^{(i)}, \tag{3}$$

where M is the number of flights of the air company AO and $S_f^{(i)}$ is the satisfaction of the i-th flight of the company.

When more than one AO is present in the system, the definition of global satisfaction is more subtle. We define two new measures of satisfaction: a *Global satisfaction* (*GS*) and a *Normalized global satisfaction* (*NGS*). Let us consider the simple case of two air companies, AO_1 and AO_2, and assume that there is a fraction f_1 of flights belonging to company AO_1 and a fraction f_2 of flights belonging to company AO_2. Finally, let us call $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ the average satisfaction of the two AOs.

The global satisfaction is the sum of the weighted satisfactions of the different air companies, where the weights are given by the corresponding portions of flights, i.e.

$$\mathcal{S}^{TOT} = f_1 \times \mathcal{S}^{(1)} + f_2 \times \mathcal{S}^{(2)}. \tag{4}$$

In the normalized global satisfaction, instead, we consider the satisfaction of an AO by normalizing it over the one computed when the "rival" company is not present.

$$\tilde{\mathcal{S}}^{TOT} = f_1 \times \frac{\times \mathcal{S}^{(1)}}{\mathcal{S}^{(1)}(f_2 = 0)} + f_2 \times \frac{\times \mathcal{S}^{(2)}}{\mathcal{S}^{(2)}(f_1 = 0)}. \tag{5}$$

$\tilde{\mathcal{S}}^{TOT}$ is sometimes a better measure of global satisfaction than $\mathcal{S}^{TOT}$ because it is insensitive to the different level of satisfaction observed when each AO is alone.

D. Using the Strategic layer as a flight plan generator

1. *Using the full Strategic Layer with a simplified interface*

The Strategic Layer can be used on its own or in conjunction with the tactical layer. In the latter case, some parameters are often not directly relevant to the tactical phase, thus in the implementation we designed a simplified interface. This simplified interface using the full strategic layer as engine, but sets most of parameters to default to "reasonable" value, loosely calibrated on real data. Hence, the user can simply input:

- A total number of flights,
- a distribution of flights per pair of entry/exit points,
- some capacities for the sectors,
- a distribution of flight levels occupancy (see section II D 3).

This simplified interface is merely a way of letting users generate trajectories easily, but the default parameters can easily be changed through optional arguments in the code.

2. Using a simplified flight plan generator

For computational reason, it was also interesting for us to write another flight plan generator. It allows to generate trajectories without considering sector capacity and is much faster than using the full strategic model.

This plan generator is designed to produce M1 surrogate trajectories starting from real data. For the considered day and for the considered airspace we generate surrogate flight plans that preserve:

- The distribution of flights between origin and destination.
- The distribution of departure times.
- The occupancy of flight levels (with odd rule, see next section II D 3).

The module requires as input the navigation point network that is generated starting from real data and produces surrogate M1 flight plans where each trajectory is the best-path on the navigation point network. The generated trajectories are therefore relative to the a non free-routing scenario, by construction. One can use the rectification procedure of section IV to generate flight plans with larger efficiency up to complete free-routing.

This module does not take into account capacity constraints. It can therefore be used in simulations where one would like to emphasize the fact that in the free-routing scenario sectors will play a minor role.

3. Generating altitudes

Both the full strategic layer – used with the simplified interface or not – and the simplified flight plan generator output 2+1d trajectories (2 space horizontal dimensions, one time dimension). However, the tactical layer is fully in 3+1d, so we have designed some automatic procedures to generate some synthetic altitudes in a comprehensive way. We implemented the following procedure: (i) we first extract a distribution of flight levels occupancy a number of flight level values equal to the number of navigation points of the considered trajectory; (ii) we then order the values so that the first third of them are in increasing order, the last third of values are in decreasing order the second third of values are mixed. This is a very simplistic procedure that nevertheless has the advantage to roughly capture the fact that aircraft have an ascending en-route and descending phase.

Different options of for the distribution of altitudes are implemented, including:

- Fully random altitudes,
- user-given analytical distribution,
- distribution extracted from data.

E. Parameters summary

In table I we produce summarize the main parameters of the Strategic Layer and the Network generator. In the third column we give a short description of the parameters and in the fourth column we introduce a classification of the parameters in terms of the three categories described below:

- FP - free parameter, to be chosen at will depending on the type of experiments one wants to perform.
- EO - external output, to be chosen according to some rule depending on the type of realistic experiments one wants to perform.

- CD - parameter that needs to be calibrated from data.
- CV - parameter that needs to be calibrated according to the validation activities performed with ATM experts and ATCOs.

TABLE I: Model parameters relevant for the generation of flight trajectories. In boldface we indicate the category associated to the parameter in the present work.

ID	**Parameter**	**Description**	**Type**	**Value**
01	N_s	Number of sectors	FP/EO	10
02	N_p	Number of navigation points per sector	FP/EO	Dis. from data (~ 28 in average)
03	$\mathcal{A}$	Area of the considered airspace	FP/EO	LIRR
04	N_a	Number of airports or entries/exits	FP/EO	189
05	$\{t_{ij}\}$	Crossing times of edges	CD	Dis. from data
06	$\{\mathcal{C}_\alpha\}$	Capacity of airports	FP/CD	Not used here
07	$\{\mathcal{C}_i\}$	Capacity of sectors	FP/CD	Dis. from data
08	d_{min}	Minimum number of navpoints between entry and exit potentially linked by a flight	FP/CD	5
9	N_{fp}	Number of flight plans submitted for each flight	FP/CV	10
10	N_{sp}	Number of paths of navigation points per path of sector	CD	2
11	τ	Time shifting step for the flight plan	FP/CV	15 min.
12	DP	Type of departure times pattern	FP/CD/EO	Dis. from data
13	δt_0	Standard deviation of noise added to the real departure times	FP	Not used here
14	N_f	Number of flights	FP/EO	Variable
15	α, β	Behavioural parameters for airlines	FP/CD	1, 0.001
16	N_{shock}	Number of sectors which are shut down	FP	Not used here

In addition to these parameters, the flight plan generator requires the specification of the distributions indicated in Table II.

TABLE II: Distributions relevant for the generation of flight trajectories.

ID	**Param.**	**Description**
1	v	distribution of the aircraft velocity
2	FL	distribution of the flight levels occupancy
3	OP	distribution of flights between origin-destination pairs
4	DEP	distribution of departure times

III. PRE-TACTICAL DE-CONFLICTING MODULE

The task of the pre-tactical de-conflicting module is to generate conflict-free planned trajectories starting from real or surrogate planned trajectories. The need for such a module is due to the fact that one of the features foreseen by SESAR will be a better planning of the trajectories such that they may be already conflict-free [9].

As such, since we are still at the planning level and no issue regarding the flight conditions is taken in consideration, differently from the modules of V F and V G, that modify the sequence of navigation points/flight levels, this module only acts on the departure time of the aircraft. In fact, we still use the sub-module that performs the Conflict Detection, see V E below. Such module is now associated with a Shift-In-Time module that simply randomly changes the departure time of an aircraft experiencing a separation minima infringement in its journey. Another big difference with the module of V is the time-step which is now fixed to 24 hours, because we want to perform such de-conflicting at a daily level. Unfortunately this requirement implies the use of a huge amount of memory and therefore the module can be computationally time-consuming[**?**].

The Conflict detection module checks for any possible conflict according to a list. If it detects a conflict it tries to shift in time the departure of the aircraft of an amount of time within the range [-5 min, 5 min]. It tries this procedure until the flight trajectory is de-conflicted, for a maximum number of 100 iterations. If at the end of the 100 iterations the aircraft is still experiencing a loss of separation it tries to shift in time the departure of this aircraft of an amount of time within the range [-10 min, 10 min], then within the range [-15 min, 15min] and finally in the range [-20 min, 20 min], if any solution is found the module starts again with another ordering of the list.

In most practical cases at the end of this process all planned trajectories will be conflict-free. Note that the pre-tactical de-conflicting model does not take into account any kind of uncertainty. It is simply designed to avoid the largest part of the conflicts by solving those which would occur if all flights would go as planned.

IV. RECTIFICATION MODULE

Our simulator will be used to perform scenario simulations in order to investigate how predictability and capacity issues will change from the current to the SESAR scenario. To this end, we decided to model the SESAR scenario as the end-point of a spectrum of possible scenarios continuously ranging from the current to the SESAR scenario.

First of all we define a new metric called efficiency E in order to measure how different a given network of routes is with respect to the situation where any pair of airports (Origin and Destination) is directly connected by straight route. We define such efficiency as:

$$E = \frac{\sum_{N_f} d(O,D)_i}{\sum_{N_f} d_{BP}(O,D)} \tag{6}$$

where $d(O,D)$ is shortest distance between Origin and Destination, while $d_{BP}(O,D)$ is the Best Path distance on the route network identified by the navigation points. The sum is over all planned flights recorded in the M1 files. In this way the more a route is congested, the more weight it has. Another possible way to have an unweighted efficiency is to extend the sum above all possible routes. This metric takes values in the range (0,1]. Of course, the value E=1 corresponds to the SESAR scenario.

A. A rectification procedure

We will move in a controllable way from the current scenario to the SESAR scenario by generating surrogate route networks each identified by a certain value of efficiency. This will be done by introducing a *rectification* of the current trajectories. This will allow us to study the transition between the current scenario and the SESAR scenario in a controlled way.

The algorithm requires as input a generic M1 file, i.e. a set of real planned trajectories, and produces as output another surrogate M1 with a larger target value of efficiency. At each step the algorithm evaluates the current efficiency and if it is less than the target efficiency performs the following steps:

- for any flight trajectory, the first and last point will not be modified, as they correspond to the origin O and destination D airports, see Fig. 6.
- then we randomly selects a navigation point P_i of a certain trajectory flight between the 2^{nd} navigation point and the one before the last navigation point.
- we substitute P_i with the medium point M_i between the previous navigation point P_{i-1} and the next one P_{i+1}, see Fig. 6. In some case, P_{i-1} might coincide with the origin O or P_{i+1} might coincide with the origin D. This is not a problem given the fact that P_{i-1} and P_{i+1} are merely used as a reference.
- this procedure is iterated until the target efficiency is reached or all navigation points of all trajectories in the M1 files are modified.

At each iteration of the above steps the procedure maintains the number of navigation points present in the M1 files.

By using such procedure we can generate a set of M1 scenarios with increasing efficiency from the current scenario to the SESAR scenario characterized by unitary efficiency. We will therefore use the model described in the previous section to generate the corresponding sets of M3 files. We will be therefore able to investigate the modifications occurring from the current to the SESAR scenario in a controlled way.

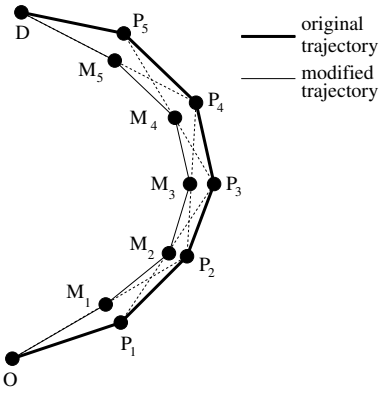

FIG. 6: The figure illustrates the techniques of rectification that we use in order to generate surrogate M1 scenarios with increasing efficiency.

B. A simplified rectification procedure

The rectification procedure described above may be rather time consuming from a computational point of view. We have therefore devised a simplified procedure that reveals to be less time consuming and therefore more appropriate when we will have to perform several sets of simulations.

The alternative rectification is done in the following way. In a first step, a point P_i is chosen at random on a trajectory, like previously. However, the point is simply removed from the trajectory instead of being moved, i.e. the flight goes from P_{i-1} to P_{i+1} directly. The procedure goes on until the target efficiency is met.

In a second time, we resample the trajectories by generating new points on the trajectory so as to ensure that the agent-based model works properly. We do this by keeping the same number of navigation points between the remaining points on the trajectories that there were originally. Hence, the number of navigation points per trajectory is kept constant too. Figure 7 illustrates the procedure.

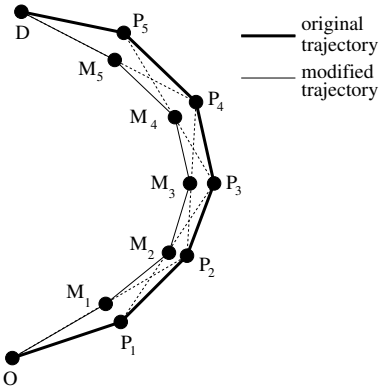

FIG. 7: The figure illustrates the simplified technique of rectification that we use in order to generate surrogate M1 scenarios with increasing efficiency.

By summarizing, our model will give us the possibility of performing simulations of the current and SESAR scenario each of them characterized by:

- **current scenario**: low efficiency, no conflict-free trajectories, look-ahead of about 20 min
- **SESAR scenario**: unitary efficiency, conflict-free trajectories, look-ahead up to 60 min

In both cases we can consider the possibility of giving or not directs ($p_d = 0$ or $p_d = 1$) and the possibility of having or not having sectors.

V. TACTICAL LAYER

The agents mainly involved in the tactical layer of our agent-based model are aircraft/pilots and air traffic controllers who are active within an *Area Control Center* (ACC) in the European airspace. We model and simulate the events that make a planned flight trajectory, recorded in the so-called M1 files, transform into an actual one, recorded in the so-called M3 files.

The interaction between the agents considered in our ABM is needed in order to manage the tactical changes occurring in the system due to unforeseen events, i.e. weather events, congestions, limitation of sectors capacity, etc. Moreover, the ATC sectors are the places where flight trajectories are made conflict free.

The model takes into account that M1 trajectories are not conflict free. Thus one main task to be performed within the model is to deconflict trajectories. Moreover, we simulate shocks in the system and see how the system reacts to them. We assume that the shock lasts for a certain time window. Operatively, this means that for a certain time window a certain area of the sector or ACC can not be crossed by flights. This might correspond to a situation where an extreme weather event occurs and therefore the air traffic must be deviated [10, 11]. As a result, another task of the model is to modify one or more flight trajectories in order to avoid these shocked areas. The way we model this step is to deviate the flight trajectories along new navigation points that are external to the restricted area and with the constraint that (i) we want to minimize the length of the deviated trajectory and (ii) the deviated trajectory must be conflict free. We will perform different simulation experiments by varying the number and extensions of shocks. However, this module will not be used hereafter.

In general we will take into account three different critical situations: (i) there is the onset of a shocking area, (ii) there is a possible conflict of trajectories that nevertheless do not intersect one with each other and (iii) there is a possible conflict of trajectories that intersect with each other. The last two cases are essentially the same from an operative point of view. We keep this case distinct from the previous one to emphasize that the last case usually occurs mainly in the planned trajectories, while the previous one usually occurs mainly when one of the two conflicting trajectories have already been deviated. In any case the way our ABM treats these three different situations is the same. Starting form the planned trajectory, we identify the navigation point(s) involved in the critical situation and try to select new navigation point(s) for each flight trajectory such that the new trajectory has the minimal length and it is conflict free. This algorithm is therefore essentially based on "re-routing" and the possibility of performing flight level changes.

When a trajectory is deviated, then in general the aircraft is sent back to its planned trajectory. The algorithm we have implemented looks for a new trajectory that allows the aircraft to come back to its planned trajectory within the considered ACC. In general an aircraft in sector S_A when re-routed can be directly sent to another sector S_B if such choice is preferable in terms of trajectory length. However, if sending the aircraft to a new sector S_B would infringe the capacity of that sector, the algorithm searches for a sub-optimal modified trajectory trying to send the aircraft back to the planned trajectory within the same sector S_A.

Whenever possible, the model allows for the issuing of directs. They are given within the ACC in order to speed up the passage of the aircraft within the ACC, provided that no conflict is created.

We have constructed the code in a modular way that allows to swap the priority of the strategies adopted by the controllers. In fact, as a default controllers first check for the possibility of doing re-routings and then change the flight altitude. However, if necessary, we can easily modify the code in such a way that the two strategies mentioned above are swapped or that, if needed, no direct is issued.

The modules described below implement a local resolution of conflicts. However, this way of solving conflicts (i) can be slow from a computational point of view and (ii) provides solutions that are not optimized at a global level, thus making it necessary to "adjust" trajectories several times as long as an aircraft travels across the ACC. We are fully aware of this limitation in our model. Indeed, we implemented such solution because we had indications that this is close to the way controllers work in reality. Moreover, we also believe that our solution might be quite effective in the SESAR scenario simulations. In fact, we might simulate a scenario where controllers have a role less preeminent than in the current scenario and some basic conflict-resolution actions are left to the single aircraft. In this respect, our model might mimic a scenario where pilots, that clearly have not a global vision of the system, endowed with a set of policy rules assigned by their airlines, will perform an *active* conflict resolution at a tactical level, thus realizing a sort of self-organization amongst aircraft.

Below we describe the different modules of our model.

A. Time-step choice

The ABM is of course a discrete-time model. Therefore the first thing to do is to introduce an efficient time discretization. In fact, we consider two typical times in the model: the time-step Δt and the elementary time-increment

δt, see Fig. 8. While the elementary time-increment δt is the one that allows a finer sampling of the trajectories, the Δt time-step is the time horizon over which the controller knows the flight trajectories to be managed.

In fact, the model works by sampling the position of each aircraft at each time t_i where $\delta t = t_{i+1} - t_i$ is typically of the order of 8 sec and there are N elementary time-increments within each time-step Δt. Therefore the time-step is of $\Delta t = N\delta t$ seconds. The aircraft travels with constant velocity from one sampling point to the successive.

Additionally, we have implemented the fact that the model works on time-steps of length Δt that are overlapping with each other. Specifically, when the calculations are performed within a time interval $[t, t + \Delta t]$, the successive time interval is $[t + t_r\Delta t, t + \Delta t + t_r\Delta t]$, where the time-roll t_r is a number between 0 and 1, see blue vertical line in Fig. 8. By considering overlapping time-steps we avoid the effect of having a controller blinded at the end of each time-step. Its look-ahead will therefore be enhanced and in that way in worst case he will see a conflict $(1 - t_r)\, N\, \delta t$ seconds in advance. In the time range $[t, t + t_r\, \Delta t]$ the controller has a perfect knowledge of the flight trajectories. In the time range $[t + t_r\, \Delta t, t + \Delta t]$ we assume that there is a degradation of the information and that, in particular, the controller has an imprecise knowledge of the aircraft velocity, see section V B.

At the successive time-step the ABM updates the starting position of each aircraft with the position observed at the time $t_r\, \Delta t$, see blue vertical line in Fig. 8.

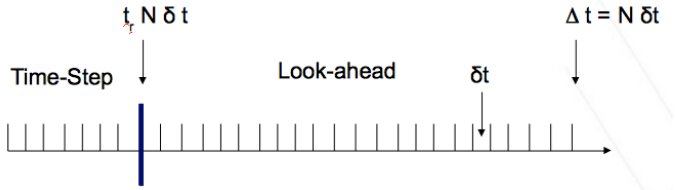

FIG. 8: Illustration of the time discretization used in the model.

B. Velocity noise submodule

This module allows to introduce some errors in the forecast of the controller. Indeed, in the most simple setup, the controller is perfect in the sense that it forecasts exactly the right trajectories for the flights within its time horizon. Hence, the only suboptimal decisions are taken because of the limited time-horizon.

In order to inject some errors on the positions of the flights, we introduce a parameter σ_V. Then for each flight we are doing the following:

- Between $t = 0$ (current time) and $t = t_r\, N\delta t$ (time of next update), we do not touch the forecast on positions. This is to ensure that there will be no actual conflict.
- Between $t = t_r\, N\delta t$ and $t = N\delta t$, we change the velocity $v \to v\,(1 + \delta v)$, where δv is drawn at random from an uniform distribution in the range $-\sigma_v$ and σ_v. Then the controller forecasts the position based on this new velocity.

The key point of this noise is the fact that the controller makes bigger errors on positions on longer horizon, and improves its forecast with time, until it is perfect below $t_r \times \Delta t$. Therefore, σ_v is not related to anything we might infer from data. Rather it is a proxy of how clever the controller is in performing reliable calculations in a short period of time to forecast the aircraft position.

We insist on the fact that the actual velocity do not change, and hence are always those of the planned trajectories (except, of course, in case of re-routing where the velocity is extrapolated on the new segment). In practice and without learning process, incorrect forecast or stochastic changes of the trajectories are indistinguishable.

C. Navigation Points

Given the ACC, we populate it with navigation points. On one hand, part of the navigation points selected are real ones, i.e., those crossed by the flights according to their M1 last-filed flight-plan. On the other hand, other navigation points are generated randomly inside the ACC from an uniform distribution. These new navigation points could be seen as temporary points (!-points) in the M3 flight-plan. However, not all of them will be really used in the flights deviations. Only a set of them will be selected, as we will explain below. All the not used ones will be eliminated from the analysis after all the flights in the ACC will be checked. As we will explain in section V F, they are generated to

allow the aircraft to deviate from the planned trajectories without necessarily passing over a predefined navigation point which might be too far.

D. Flight list submodule

Once the ACC has been populated with navigation points, we create a list FL_k of flights active in a specific time-step in the considered ACC. Such list will be reshuffled in the next time-step. This means that at each time step we randomly reorder the aircraft and therefore the order by which the aircraft are taken into account for conflict detection changes in each time-step. This is done in order to be sure that the trajectory to be deviated is not always the same one.

We have also the possibility of reordering this list according to a pre-specified rule or a scoring system. This option might be useful in the SESAR scenario as a way to implement different airline policies. For example, one might put in the higher part of the list those aircraft belonging to airlines prone to accept trajectory changes because in doing so they pay discounted fees to the air traffic providers. Analogously, one might put in the lower part of the list aircraft belonging to airlines less favorable to accept trajectory changes because they want to maximize predictability.

Within this list we check whether the flights are crossing a shocked area and whether or not they conflict with each other. Specifically, the i-th aircraft in the list will be checked against all other $i-1$ flights with $j < i$. However, not all the $i-1$ flights might have an interaction with i-th flight within a certain time-step. In fact, if the distance between the starting position of the two i, j flights is larger than $2\, v_{max} N\, \delta t + d_{thr}$ it is impossible that the i-th and j-th flights will be involved in a safety event, see Fig. 9. Here d_{thr} is the typical distance threshold for safety events usually considered to be equal to 5 NM and v_{max} is the largest velocity of the aircraft active in the ACC. For this reason at each time-step the ABM first selects among the $i-1$ flights with $j < i$ those who have possible conflicts according to the rule indicated above and then checks for conflicts only those in such subset. When a trajectory modification is needed, it will affect the i-th flight.

The reason for shuffling the list at each time step Δt is in order to avoid that the trajectory modifications are always applied to the same aircraft.

If a conflict involving the $i-th$ is not solved by one of the strategies mentioned below, then the list is modified by putting the $i-th$ in the first position in the list and trying to modify the other trajectories.

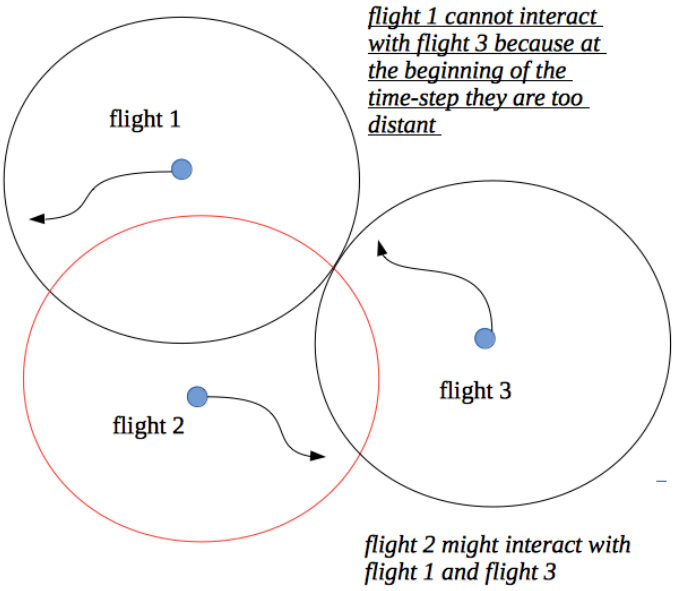

FIG. 9: Illustration of the procedure we use to optimize the procedure we use to check flight trajectories done against the other.

E. Conflict detection module

In order to check for collisions between two flights, we use a data structure that considers the aircraft localized inside the trajectory segments travelled within the given time-step Δt. For this purpose, as mentioned above, we

introduce a finer subdivision of the time-step into N elementary time-increments δt and compute the real space-time position of the aircraft at each elementary time-increment, by assuming a constant velocity. The collision algorithm will have to simply calculate the positions of the aircraft for each of the elementary time-increments and then compute the distances between the two aircraft at these positions. Suppose we are now checking if the i-th flight trajectory is conflicting with all other f_j trajectories, with $j < i$, as mentioned above. We are therefore considering a maximal number of $i-1$ flight trajectories. For each of them we have an array $\mathcal{P}_j$, $j = 1, \cdots i-1$ of length N given by the positions computed according to the algorithm illustrated above. For each of the N elementary time-increments, we compute an array of distances $d^{(i)}$ of length $i-1$ choosing as a value for each array element the minimum Haversine distance [12] between the i-th aircraft and all the other $i-1$ aircraft in the list with $j < i$. A possible conflict between two aircraft is detected whenever the elements of such an array are smaller than the safety distance threshold d_{thr} that is usually set to 5 NM. When a conflict is detected the algorithm proceeds to the next module that performs the de-conflicting of trajectories.

F. Conflict resolution submodule

After the check for collision has been done, this module searches for a new conflict-free trajectory. It is conceived as a three-step algorithm that acts on the search of a new trajectory (re-routing) and the change of flight level, in case conflicts exist. The order by which these steps are applied might be changed.

The conflict resolution module is trajectory-based and performs a local resolution of conflicts by evaluating each pair of conflicting trajectories. The module works in 3-D as it takes into account the different flight levels at which aircraft are and issues flight level changes if needed.

The first step of the module we present here is the one that performs the re-routing. The procedure is illustrated in Fig. 10. We first identify the point B (not necessarily a navigation point) at which the aircraft is when the considered time-step begins. We then identify the navigation point A which is the first navigation point after the collision or the shocked area (filled circle in the figure). The idea is to (i) keep B, (ii) substitute A with a temporary navigation point and (iii) in case of conflicts eliminate all the navigation points between B and A of the planned trajectory within the considered time-step Δt and in case of shocked areas eliminate all subsequent navigation points inside the shocked area. Let us call E the first navigation point in the planned trajectory that we can keep.

To solve the conflict, we take the previously generated temporary navigation points T_k (squares in the figure) falling in a circle of radius D_{max} centered in B. We order them with respect to the angle that the segment connecting B and T_k forms with the original trajectory. We select the temporary navigation points that have an angle smaller than a certain critical value α_M, see Table III. We define as a new trajectory the one given by the two segments between B and T_k and between T_k and E. Amongst all the admittable T_k navigation points we select the one for which (i) the deviated trajectory has the smallest length and (ii) the angle between the planned trajectory and the segment between T_k and E is smaller than α_M.

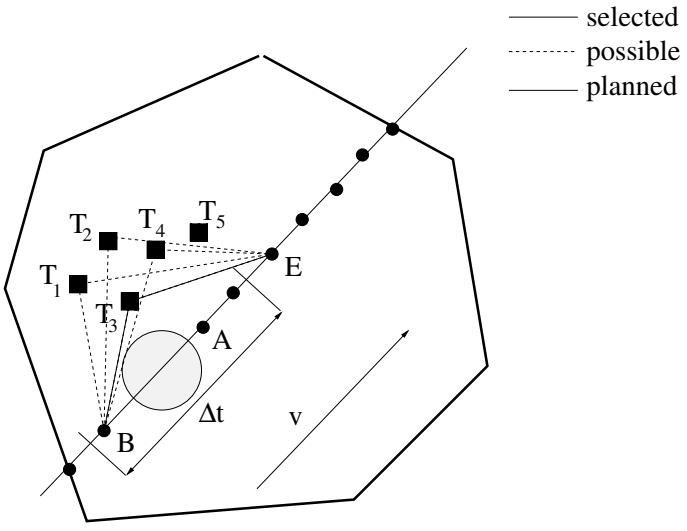

FIG. 10: The figure illustrates the procedure of re-routing, see text for more details. The dashed trajectories, although possible, were not selected because the angles between the modified and planned trajectories were larger than α_M. T_5 was not considered because the segment BT_5 would cross the shocked area.

If we do not find any temporary navigation point fulfilling these requirements we substitute E with the successive navigation point until the temporal distance between A and E is smaller than T_{max}. If no solution is found then the algorithm exits this sub-module and go to the next one.

When a trajectory is deviated, then in general the aircraft is sent back to its planned trajectory. The algorithm we have implemented looks for a new trajectory that allows the aircraft to come back to its planned trajectory within the considered ACC. In general an aircraft in an sector S_A when re-routed can be directly sent to another sector S_B if such choice is preferable in terms of trajectory length. However, if sending the aircraft to another sector S_B would infringe the capacity of that sector, the algorithm searches for a sub-optimal modified trajectory trying to send the aircraft back to the planned trajectory within the same sector S_A.

G. Conflict Resolution submodule - Flight level change submodule

The second step of this module involves changes of flight level. A Flight level (FL) is defined as altitude above sea-level in 100 feet units measured according to a standard atmosphere but allowed flight levels are separated by 1000 feet, thus in principle 10 flight levels. However, according to the standard practice, in the following, a change of one FL is to be intended as a variation of 1000 feet in altitude (one separation level). Moreover, the semicircular rule has been considered, meaning that aircraft flying in opposite directions are allowed to fly only to odd or even FL respectively. Therefore when an aircraft needs to be moved to another FL, it will not be moved to the next first one but to the second one to respect the semicircular rule.

All flights are initially considered to be active in the planned M1 flight level. Therefore they can move two Flight Levels (FL) upwards or downwards whenever the re-routing is not feasible. The choice of the new level is done by considering the one where there is a smaller number of potential conflicts. If no level is available then the list is reshuffled by moving the considered flight in the first position.

As in the previous case, the flight is sent back to its trajectory within a time T_{max}.

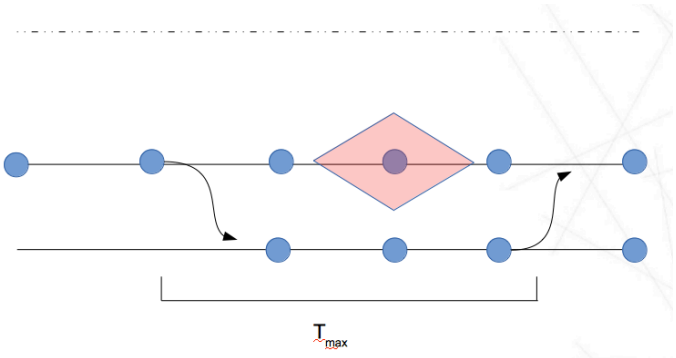

FIG. 11: Illustration of the flight-level change sub-module.

H. Directs submodule

If any conflict is detected, the ABM tries, with a probability p_d, to give a direct to the i-th flight, see Table III. A direct is made by removing one or more navigation points of the M1 flight-plan after the first navigation point present in the current time-step.

The algorithm first evaluates how many navigation points can be removed with the constraint that the flight has to come back on the original route within a time interval equal to $T = 2\,\Delta t$, without infringing the sectors' capacities. Such choice of T has been done in agreement with the indications of the air traffic controllers consulted within the ELSA project. After that the model evaluates if the new route will be involved in conflicts (with other flights or shocks). If the direct is safe the new route is accepted, otherwise no direct is issued.

Finally, we have implemented a sensitivity threshold L_s. If the absolute difference between the length of the planned trajectory and the trajectory modified with the introduction of the direct is smaller than L_s, then we do not consider such direct. The threshold is set to $L_s = 1$ km. This is done in order to avoid a proliferation of directs that have negligible impact on the improvement of the system efficiency.

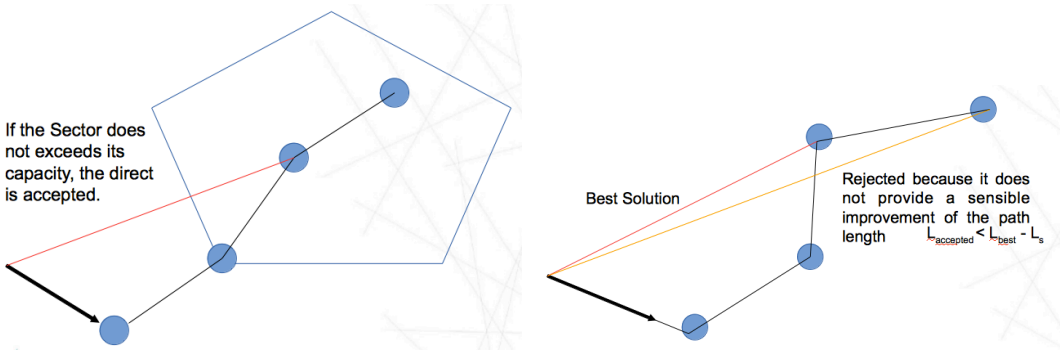

FIG. 12: The figure illustrates the procedure of issuing directs.

I. Shocks submodule

In the current version of the ABM the shocks are modeled as circles of center C_S and fixed radius R_S and located at a flight level drawn from a random uniform distribution in the range $[FL_{min}, FL_{max}]$. Each shocks vertically extends over 1 separation level, i.e. 1000 feet. Each shock has a duration drawn from a random uniform distribution in the range $[1, D_S\, N\, \delta t\, t_r]$. The area within these circles is inaccessible for the aircraft, and if a shock appears along an aircraft flight trajectory, the aircraft has to be re-routed or change flight level because all maneuvers are interdicted inside the shocks. We implemented the fact that there is an average number S_m of shocks per time-step per flight-level. In our model, the number of shocks will follow a Poisson distribution with mean S_m.

The position of the shocks is drawn from a list of points provided by the user. In this way the user can obtain a uniform distribution of the shocks inside the ACC providing a uniform distribution of the points C_S, or he/she can obtain different spatial distributions by providing an appropriate list.

At the beginning of each time-step the controllers cannot forecast the shocks. This means that they look at the current position of the shocks and they operate assuming that the shocks are fixed along the time-step Δt even if they could disappear within the $t_r \Delta t$ time horizon. We can illustrate this issue with an example. Suppose we are considering an aircraft i at the time-step $R_1 = [t, t + \Delta t]$. As mentioned above, let us first consider the time interval $R_{1,exact} = [t, t + t_r\, \Delta t]$. This is the time range where the aircraft positions is known exactly. The controller has to decide what to do with the aircraft trajectory in the time range $R_{1,l-a} = [t + t_r\, \Delta t, t + \Delta t]$, which corresponds to the controller's look-ahead. Since shocks are issued every Δt seconds, all shocks occurring in the time range $R_{1,l-a}$ will be known by the controller not in the time-step R_1, but in the successive one $R_2 = [t + t_r \Delta t, t + t_r \Delta t + \Delta t]$. Therefore in the time-step R_1 the controller manages the trajectory by assuming that even in $R_{1,l-a}$ there are no shocks.

J. Multi-sector submodule

The ACC we are considering is divided into a number of sectors. Each sector is characterized by its geographical extension and by its capacity, here intended as the maximum number of aircraft that can be contemporarily present in the sector within a certain time window.

The way we implement capacity in the sectors is by assigning a flag to each navigation point of the M1 files present in the sector. The flag indicates the sector a navigation point belongs to. We assume that the number of flights that can travel over each navigation point in a certain time-window is given by the sector's capacity. All navigation points on the boundaries between sectors have associated a zero flag. This value identifies the fact that no capacity constraint is assumed to exist on the sectors boundaries.

At each time-step the ABM evaluates the occupancy of each sector of the ACC. The occupancy is defined as the number of flights that will cross the sector within an hour.

When the occupancy of sector exceeds its capacity all the re-routings or directs that come from other sectors are rejected. Operatively this means that in a condition when occupancy equals or exceeds capacity any other incoming flight has to enter the sector from the M1 planned navigation points, see Fig. 13.

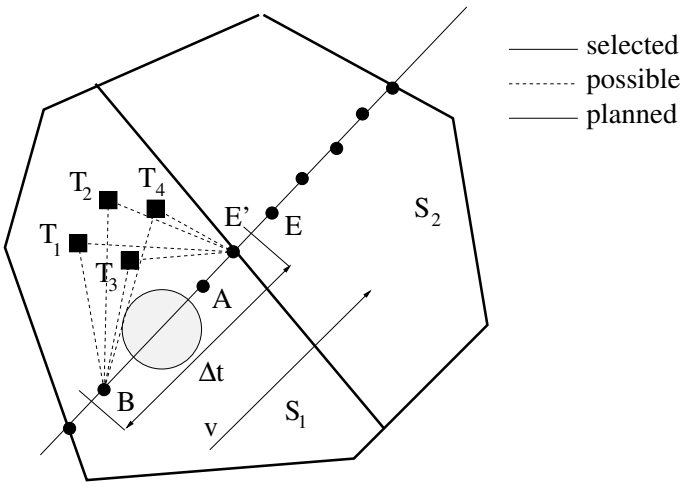

FIG. 13: The figure illustrates the techniques of re-routing when the ACC is splitted in sectors, see text for more details. When the capacity of sector S_2 is exceeded, the module redirects the aircraft to the navigation point E' rather than E as in Fig. 10.

In the present version of the model, the controllers behave in the same way, i.e. the models parameters summarized in Table III are the same for controllers in the different sectors. This is done in view of the fact that at the level of ACC there is a certain uniformity of behavior amongst the controllers of different sectors, also due to the fact that they are all working in the same physical place. This also contributes to have a lower level of coordination amongst controller, with respect to the case of controllers working on sectors belonging to different ACCs. Of course, when the model is used at the level of the ECAC airspace, one should implement that controllers might have different behaviors and therefore the model's parameters should be selected in a different way for different controllers.

TABLE III: Model parameters relevant for the management of flight trajectories.

ID	Parameter	Description	Type	Value
01	Δt	Length of the time-step. This is also related to the controller's look-ahead.	FP	12.27m
02	δt	Length of the elementary time-intervals.	FP	8s
03	t_r	Fraction of Δt by which we move the overlapping time-steps.	FP	0.25
04	σ_v	Range of the noise introduced in the estimation of the aircraft velocity	CV	not used here
05	D_{max}	Radius of the circle centered in B where we look for temporary navigation points potentially relevant for performing a re-routing.	FP	not used here
06	α_M	Maximum angle of deviation between planned and modified trajectory.	CV	15 deg
07	T_{max}	Maximal temporal distance between the navigation point B and navigation point E that identify when a deviated portion of flight trajectory starts and ends.	FP/CV	24.5m
08	p_d	Probability to try to issue a direct.	CD/CV	0
09	L_s	Sensitivity threshold for issuing a direct.	FP/CV	not used here
10	C_S	Center of each shock.	FP/CD	not used here
11	S_m	Average number of shocks per time-step per flight-level.	FP/CD	not used here
12	D_S	Temporal duration of each shock.	FP/CD	not used here
13	R_S	Radius of each shock.	FP/CD	not used here
14	$FL_{min/max}$	Minimum/maximum flight level where shocks are generated.	FP/CD	not used here

VI. POST-PROCESSING

The repository also features some post-processing tools which help the user having a better grasp on the results of the model. The functions are currently scattered in several files, but include for instance network-builders from trajectories, computation of different network metrics, comparisons of networks, etc. A movie maker is also available

to see the behaviour of the model in real time.

---

[1] https://github.com/ELSA-Project/ELSA-ABM.

[2] https://github.com/ELSA-Project/ELSA-ABM/blob/master/doc/Get%20Started.pdf.

[3] https://github.com/ELSA-Project/ELSA-ABM/blob/master/doc/ELSA_E.02.18_WP2deliverable_D2.3.pdf.

[4] https://github.com/ELSA-Project/ELSA-ABM/blob/master/doc/ELSA_E.02.18_WP2deliverable_D2.4.pdf.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, dec 1959.

[6] J Y Yen. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, 1971.

[7] B Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[8] A Rényi P. Erds. On the Evolution of Random Graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[9] EUROCONTROL. SESAR Concept of Operations Step 1. page 146, 2012.

[10] Adrian Agogino and Kagan Tumer. Regulating air traffic flow with coupled agents. In *Proceedings of the 7th Int. Conference on Autonomous Agents and Multiagent Systems*, number Aamas, pages 535–542, Estoril, 2008.

[11] S.R. Wolfe, Peter A. Jarvis, F.Y. Enomoto, Maarten Sierhuis, B.-J. van Putten, and K.S. Sheth. A Multi-Agent Simulation of Collaborative Air Traffic Flow Management. In Ana Bazzan and Franziska Klügl, editors, *Edited Collection on Multi-Agent Systems for Traffic and Transportation*, pages 357–381. IGI Global, jan 2009.

[12] W. M Smart. *Text-Book on Spherical Astronomy*. Cambridge University Press, 6th ed. ca edition, 1960.

[13] Note that this is actually more complicated than that. In reality, since the restriction can include shorter sector paths, we have to cut the edges between two navigation points which are not in two consecutive sectors.

[14] This parameter is usually set to 15 or 20 minutes, which is close to the typical duration of the airport slot.

[15] Indeed, it is a brute-force method which seeks a minimum to the total number of conflicts, the variables being the time of departure of the flights.