Adaptive Air Traffic with Big Data Analysis

Arcady Rantrua, Sébastien Chabrier, Marie-Pierre Gleizes

Adaptive Air Traffic with Big Data Analysis

Arcady Rantrua
Sopra Steria / IRIT
10 Rue Claude-Marie Perroud, 31100 Toulouse, France
118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9, France
arcady.rantrua@soprasteria.com, arcady.rantrua@irit.fr

Sébastien Chabrier
Sopra Steria
8 avenue Yves Brunaud, 31770 Colomiers, France
sebastien.chabrier@soprasteria.com

Marie-Pierre Gleizes
IRIT
118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9, France
marie-pierre.gleizes@irit.fr

ABSTRACT

Air traffic generators are widely used in the air traffic control and aeronautics industry for training purposes and to validate new concepts or new systems. But it is difficult for a machine to generate realistic trajectories and behavior for an aircraft which lead to the participation of many humans in the simulation to compensate. Traditional generators require performance data for aircrafts and heavy preparation to generate realistic trajectories. Our approach takes advantage of widespread flight location data to learn from their behavior and improve traffic generation. Combining cooperative multi-agent learning methods and big data we show that it is possible to design and generate a realistic and intelligent traffic more easily and without human intervention.

Keywords: simulation, multi-agent system, decision making, air traffic management, machine learning, data value

1 INTRODUCTION

Simulation of air traffic is broadly used in the air traffic control world for the training of pilots or controllers, the validation of new functions or systems, and demonstration purposes.

Those platforms put controllers and pilots in as-real-as-possible situations to test their ability to take the good decisions in real-time. For example, for a controller it might be useful to evaluate if he or she is able to know when to give an order to a pilot and what order to give. That is why the simulation has to be realistic, otherwise the tests would be meaningless.

In this paper we will first present the state of the art of the simulation platforms, their limits and why a new approach is possible and needed. In the second section we will explain what kind of data we use and how we use them. Then we will present our approach based on cooperative multi-agent systems [1], our results and analysis.

2 Current air traffic simulation methods

In the real world, an aircraft has a flight plan (a list of waypoints it will overfly). But it never happens as planned. Events are part of an aircraft's life:

- the company delayed the departure time,
- a controller intervening to prevent aircrafts from being too close to each other,
- a controller giving permission to the pilot to use a shortcut,
- a meteorological event forbid any aircraft to go through a certain area.

The way an aircraft flies depends on a lot of variables: speed, climb rate, weight, wind … Current air traffic generator uses those variables to build physical models of the aircraft and, with the flight plan, can generate a trajectory.

The technical specifications of those aircraft are collected [2] to be used in a kinematic model. Those aircrafts then follow the scenario of the simulation which is crafted by hand to match the specifications of the current exercise. Sadly, every simulation with the same trajectories will be the same because those simulated aircrafts don't have any decision making abilities: they cannot adapt to a new situation the person in training could (and will) produce. It is possible, and that is the current way of solving this problem, to add other human pilots and controllers to enhance the simulation but it is a heavy and costly process.

2.1 Existing platforms

In this section we present a list of existing simulation platforms. This list is not exhaustive but is sufficient to grasp what kind of problem and limitations exist in the field.

RATSG [3] is a MIT software able to generate 4D (latitude, longitude, altitude and time) trajectories passing by navigation points. It generates changes in trajectory, speed and other parameters based on the technical specifications of the aircraft and basic mechanics equations. It is used in real-time simulation with human in the loop and could, according to its authors, be used in accelerated simulation. The scenario can include complex elements such as the triggering of audio recordings and can force aircrafts to be in a conflicting state (common in training simulation).

AirTrafficFX [4] is a real-time traffic generator for Microsoft Flight Simulator. It can generate in-flight traffic and specific aircraft formations. Scenarios must be very precise (the full flight plan has to be specified) and it needs humans to adapt to changes when running.

eATG (enhanced Air Traffic Generator) [2] is a traffic generator developed by Eurocontrol using the specifications available in the BADA database. Since this database is completed by aircraft manufacturers themselves, it is almost always up to date and covers

98% of the aircrafts flying in the European airspace. Trajectories of aircrafts are computed based on the flight plan, the model of the aircraft using variable mass system equations. Humans are integrated into the simulation and that is how the system takes into account the changes coming from the controller.

Modifying existing traffic is also a solution to "generate" traffic. A good example of this method is the one used at ACT-250 [5]. They modify recording in two different ways. First, they are able to increase traffic load by doing *temporal compression* which means reducing the minimum time between two planes taking off. Second, they do random time adjustments to flights to the time of departure of the flight.

2.2 Limits of current approaches

As we saw in the previous section, existing simulators exclusively model aircrafts based on their technical specifications and those specifications are not easily available. And even if they were, modeling the aircraft is not enough: the aircraft has a pilot who sometimes takes a shortcut, or goes a little faster, sometimes there are unwritten habits about how to navigate through a sector, and sometimes the rule is written but not available for everyone to see. There is much more to simulate than the aircraft: the whole environment must be simulated and there is no specification for that.

Modifying existing traffic is, at least in part, a solution. The global traffic looks different and it might enough in some cases but trajectories are copies of existing trajectories and the consistency of the traffic is comprised.

With so many actors taking decisions and so many rules it is not possible for humans to build a reliable model: we think learning is the way to go and we have a huge amount to go through to get realistic models.

2.3 The EVAA approach

We feel we can improve this process by using real flight data of many flights and during a large period of time. Every civilian aircraft has a transponder that broadcasts information such as its identifier and position and we can use that information to build a model of its behavior. It would be a huge improvement for the domain since it would enable users to make realistic simulations much more easily, more frequently and with more realism.

We feed the learning algorithm of EVAA with real-flight data to obtain a model of how aircrafts behave in the sky. With this model we can create simulation where aircraft agents are capable of making decision based on the situation we observed during the learning. It means we could have virtual aircrafts flying from point A to point B without having to specify anything other than the actual coordinates of A and B. And then, if the scenario forces the aircraft to go a specific way the aircraft will adapt its trajectory and behavior in a realistic way.

3 AUTOMATIC GENERATION OF TRAJECTORIES

Every month there is policy changes to the air space; simulator must be capable to take into account those changes which means we have to be able to treat the data. We will see in the first subsection the amount of data that our system has to take into account. And, finally, we will describe the algorithm itself.

3.1 Data volume and format

To power our algorithm we take multiple samples of flight data in a specific zone for a long period of time. Some data are very dynamic like location (latitude, longitude, altitude), speed, heading. They are sampled every four seconds which is a common refresh rate in this domain. Others are static data (unique for a single flight) like flight number, aircraft manufacturer and model, airline.

A sample is a combination of static and dynamic data in the form of a tuple of variable where each attribute is indicative of the aircraft's state at a specific time. We store those samples in a database which will be used during the offline learning phase.

Table 1 shows the **volume** and **velocity** of those data according to several parameters.

Table 1- Volume of data available for learning

Zone	Duration	Points	Flights	Size (GB)
Toulouse – Paris Orly	1 day	119 739	59	0.02
Toulouse – Paris Orly	1 month	3 652 040	1 813	0.14
France	1 day	16 200 790	13 675	2.41
France	1 month	494 124 095	417 074	73.64
World	1 day	322 374 231	80 000	55.58
World	1 month	9 832 414 038	2 440 000	1 695.10

The "Points" column is the number of samples made for this zone and duration. The "*Flights" column* is the number of recorded flights for this zone and duration.

Since the air traffic controlled policies can be changed every month, there could be many differences between recordings from two consecutive months. We plan to apply our system to monthly data to extract behaviors and policies.

3.2 Learning algorithm

We use multi-agent learning. The main principle of multi-agent systems is to use the agent to have a different grasp of the problem at hand [6]. We treat the data using an approach based on a self-organizing multi-agent architecture [7]. The aim of our learning algorithm is to be able to generate easily aircraft realistic trajectories in a simulated environment. We want to learn how an aircraft behaves and to be able to reproduce a similar behavior.

The recorded flight data are replayed and observed by the learning algorithm in two steps:

- The first step is learning on a single flight. An aircraft agent observes the aircraft and stores relevant information thanks to *point of interest* agents
- In the second step, we consider multiple trajectories; the points of interest agents works together to build a graph that will, in the end, represent flights behaviors.

We describe more precisely how this process works in the following sections.

Two-point trajectory

First let's review an example of a recording showing an aircraft going in a straight line from point A to point B. When the aircraft is in A the algorithm will memorize this location as a situation of interest (SA). We define a *situation* as a set of variables that represents the current state of the aircraft. When the aircraft finally reaches B its current situation has changed to SB. The system will then conclude that "an aircraft in a situation similar to A should aim to be in a situation similar to B". We call this simple rule a *situation vector*.

It means that the system is given a pair of item (SA, SB) where SA is the current situation and SB is what we want our system to predict.

Multi-point trajectory

After this simple example let's see what happens in the case of a complete trajectory of a single aircraft. The aircraft doesn't fly in a straight line because it follows waypoints defined by air traffic control organism. Those waypoints are crossing points for aircrafts; it's basically like an intersection for cars. The recording will show that the aircraft follows a trajectory in straight lines (segments) between waypoints, and in curved lines around waypoints.

When the aircraft turns (left or right), it means that there is a *situation of interest* at the current location. A point of interest is placed here and another will be placed the next time the aircraft turns, defining a segment. We apply for each segment of the trajectory the algorithm previously described: we obtain an ordered list of the points the aircraft successively went through.

From turn to situation shift detection

We detect turns because the resulting points provide a good summary of the route used by the aircraft. But an aircraft moves in a three dimensional space: we have to take into account latitude, longitude and altitude to detect the "turns" of the aircraft.

What about a change of speed? Or any other relevant parameter? We have to add it to the list of parameters that defines a situation. It is now difficult to keep calling that "turns detection" and we decided to call that a *situation-shift detection*.

Multiple trajectories

We described how the first step (on a single aircraft) of the algorithm works. In this section we will see how that information is aggregated to extract powerful and reliable behaviors.

The recording can show multiple aircrafts flying, turning and crossing paths. Sometime they have a *situation shift* around the same geographical location even if their trajectories diverge completely. It means that in this precise location we may have to discriminate between multiple situation vectors. *Aggregated points* are created to merge those data. The result of our learning algorithm is a graph where each node is an aggregated point and each edge is a situation vector. Those vectors can are used to direct a plane from one point to another, creating a trajectory.

4 RESULTS

In this section we compare real trajectories to simulated trajectories. First, we show that the simulation give realistic and various trajectories. Then we show an example where the learning converges.

4.1 Relevance and realism

Figure 1 and 2 present the trajectories of recorded and simulated aircrafts going from Toulouse to Paris. We can see that the simulated trajectories (Figure 2) are close to the original one (Figure 1) we used for learning but not exactly the same. It means that we do not copy what we have learned, we use the graph from the learning phase (see 3.2) to obtain an autonomous aircraft able to decide where it should go next depending on it current situation.

Figure 1: raw data on which the learning was made

Figure 2: simulated trajectories

The amount of behavioral data we detected can be represented by the amount of *situation vectors* that we found and the amount of behavioral data actually learned can be represented by the number of *aggregated points* that we stored. On Figure 3 we can see that the number of situation vectors grows linearly with the number of aircraft in the simulation and that the number of aggregated point grows logarithmically and even stabilize at the end when the system learned all it could on the dataset. We stopped the graph at 30 aircrafts for readability but it is important to the number of situation vectors keep growing linearly and that the number of aggregated points stays constant.

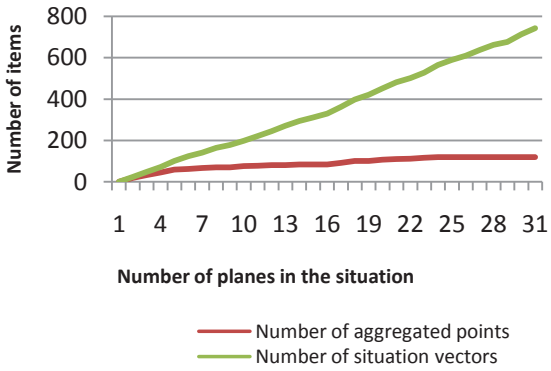

Figure 3 - Number of situations vector and aggregated points on Toulouse-Paris flights

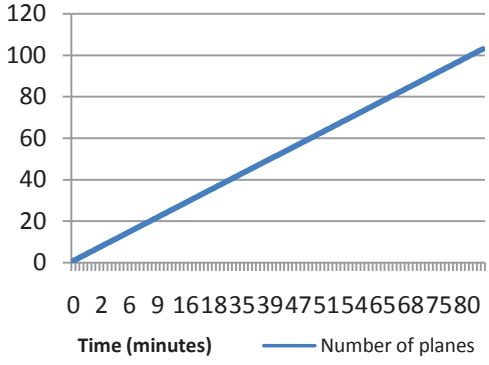

Figure 4 - Simulation duration according to the number of planes simulated

4.2 Performance

Considering the amount of data we have and how often the air traffic control policies change we need an algorithm with good performance. To evaluate performance we measured how much time the learning was taking according to the number of plane simulated. The tests are made on a laptop and the results are shown in Figure 4.

There are approximately 60 planes a day on the route we studied and we learn the behavior of those planes in 47 minutes. It might not be fast enough if we want to apply our system to the full world but it is possible to apply it to a country. We can also see that the time to complete the simulation increase linearly with the number of plane simulated which is a good indication of scalability.

5 CONCLUSION

We saw in the previous section that our algorithm delivered realistic behaviors for aircrafts in terms of trajectories and flight duration. We also noticed that the learning time increases linearly with the size of the data sample proving that our system is scalable. This is because agents work from a local point of view. It is opposed to a global approach with a global evaluation function that would see the sample size increase much faster. The reader can see a video example of the process [8] shows the different phases from recording to simulation.

This article presents a new approach for generating smart air traffic: a traffic in which aircrafts don't just fly as defined in their flight plan and don't copy another flight. We generate traffic that has never been seen with very little specification necessary because aircraft agents are capable of automatically making decision. We saw in the state of the art that this is an important improvement on the existing simulations of air traffic in order to improve aided-decision for controllers and pilots. Then, we described the big-data element of our research by presenting what kind and volume of data we had to treat. Finally, we presented how our learning algorithm worked and presented results that showed its relevance, realism and scalability.

In future works we plan to gain access to more input data like weather and to be more resistant to noise in the raw data. We also plan to extend the scope of what kind of behavior we can learn: trains, cars, humans …

6 REFERENCES

1. L Panait, S Luke "Cooperative Multi-Agent Learning : The State of the Art", Autonomous Agents and Multi-Agent Systems 11.3, p. 387–434, 2005.
2. S Gillet, A Nuic, V Mouillet, "Enhancement in realism of ATC simulation by improving aircraft behaviour models", 2010.
3. J Hansman, "RATSG : Robust Air Traffic Situation Generator", https://web.archive.org/web/20120702160227/http://web.mit.edu/aeroastro/www/labs/AATT/reviews/ratsg.html, 1996/
4. Light One Software, "AirTrafficFX", http://www.flight1.com/products.asp?product=airtrfcfx, 2009.
5. R D Oaks, M Paglione, "Generation of Realistic Air Traffic Scenarios Based on Recorded Field Data", Annual ATC Association Fall Conference, T. 46, p. 142, 2001
6. J Ferber, "Multi-agent systems: an introduction to distributed artificial intelligence", Addison-Wesley, 1999.
7. G Di Marzo Serugendo, M Gleizes, A Karageorgos, "Self-Organisation and Emergence in MAS: An Overview", Informatica (Slovenia) 30.1, p. 45–54, 2006
8. A Rantrua, Video of EVAA in action, http://www.irit.fr/~Arcady.Rantrua/EVAA/evaa_icdsst2015.mp4, 2015