Fourteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)

Towards Conflict Resolution with Deep Multi-Agent Reinforcement Learning

Ralvi Isufaj
Logistic and Aeronautics Group
Autonomous University of Barcelona
Sabadell, Spain
ralvi.isufaj@uab.cat

David Aranega Sebastia
Centre de Recerca Matemática
Autonomous University of Barcelona
Bellaterra, Spain
david.aranegas@e-campus.uab.cat

Miquel Angel Piera
Logistic and Aeronautics Group
Autonomous University of Barcelona
Sabadell, Spain
miquelangel.piera@uab.cat

***Abstract*—Safety in ATM at the tactical level is ensured by human controllers. Automatic Detection and Resolution (CD&R) tools are one way to assist controllers in their tasks. However, the majority of existing methods do not account for factors that can affect the quality and efficiency of resolutions. Furthermore, future challenges such as sustainability and the environmental impact of aviation must be tackled. In this work, we propose an innovative approach to pairwise conflict resolution, by modelling it as a Multi-Agent Reinforcement Learning (MARL) to improve the quality of resolutions based on a combination of several factors. We use Multi-Agent Deep Deterministic Policy Gradient (MADDPG) to generate resolution maneuvers. We propose a reward function that besides solving the conflicts attempts to optimize the resolutions in terms of time, fuel consumption and airspace complexity. The models are evaluated on real traffic, with a data augmentation technique utilized to increase the variance of conflict geometries. We achieve promising results with a resolution rate of 93%, without the agents having any previous knowledge of the dynamics of the environment. Furthermore, the agents seem to be able to learn some desirable behaviors such as preferring small heading changes to solve conflicts in one time step. Nevertheless, the non-stationarity of the environment makes the learning procedure non-trivial. We argue ways that tangible qualities such as resolution rate and intangible qualities such as resolution acceptability and explainability can be improved.**

***Keywords*—Conflict Resolution; Air Traffic Management; Reinforcement Learning; Multi-Agent Deep Deterministic Policy Gradient**

I. Introduction

The mission of air traffic management (ATM) is to make air traffic possible by means of efficient, environmentally friendly and socially valuable systems [1], [2]. At the heart of the current ATM system at the tactical level are human air traffic controllers (ATCo) who control airspace units known as sectors. The main duty of ATCos is to guarantee safety, which is accomplished by communicating and issuing instructions to pilots, monitoring traffic to maintain safety distances etc. The ability of controllers to guarantee efficient solutions that include different quality factors is constrained by their workload which can be defined as the mental and physical effort required to manage traffic [3].

In the future, challenges such as sustainability, the environmental impact and fuel consumption will have to be tackled. As a result, these factors must be also accounted in conflict resolution, to not only solve them but to assure efficiency and quality in the resolutions. Conflict detection and resolution (CD&R) tools are a way to assist ATCos in their conflict resolution duties. Conflict resolution algorithms have been a prominent research within the ATM community, with many models being proposed. For a comprehensive review we refer the reader to [4]. Early works, such as [5] prescribe fixed maneuvers for particular geometries to conflict aircraft. However such approaches are not preferred as they are not flexible and result in inefficient resolutions. In more recent approaches, various mathematical formulations are used to calculate more efficient maneuvers. For instance, Pallottino et al. [6] employ mixed integer programming to solve conflicts where they consider speed and heading changes separately. However speed changes alone cannot solve all conflicts (e.g. head-on situation). Furthermore, they consider immediate heading changes, which is not realistic. The Model Voltage Potential (MVP) algorithm proposed by Hoekstra [7] considers airspace as a potential field and aircraft as particles navigating it. The predicted future positions of conflict aircraft at the closest point of approach (CPA) are used to repel each other and thus displacing the predicted positions at CPA. The avoidance vector is calculated as the vector starting at the future positions and ending at the edge of the intruder's protected zone. Peyronne et al. [8] use b-splines to smoothly and minimally change trajectories to solve conflicts at the tactical level. Their approach, however, has several limitations, as they assume constant speed of aircraft and evaluate their approach only on academic examples. The approach proposed in [9] uses probability reach sets to represent aircraft locations and resolution is performed by separating these sets. This model suffers when the number of present aircraft is increased and they only consider that aircraft will be at their intermediate waypoint of the resolution maneuver at the same time. A multi-agent approach is considered by Breil et al. [10]. There, they model each conflict aircraft as an agent, which has to solve its local conflicts through speed changes. First of all, as previously mentioned, not all conflicts can be solved only through speed changes. Furthermore, an agent in this approach can only choose to cruise, accelerate or decelerate at fixed rates, which is not flexible and can lead to inefficient solutions. They extend their approach to also include heading changes, but also there they only allow for fixed changes.

While these approaches are successful, the majority of them have limits due to the assumptions they make about the airspace. Most discretize space to maintain computational feasibility, which means that they have fixed maneuvers that they issue. This lowers the flexibility of the solutions which may lead to inefficiencies in terms of quality of the resolution. Whereas a method that can handle a bigger or continuous resolution space could find more efficient solutions. Furthermore, most algorithms make some assumptions about the dynamics of the environment causing such methods to fail when faced with conflicts that do not adhere to those assumptions. Finally, a majority of existing methods optimize only for resolution, and might not consider the effects of the solution on surrounding traffic, fuel consumption etc.

Reinforcement learning (RL) and especially deep reinforcement learning (DRL) have recently emerged as a very successful method to tackle decision-making problems. They have achieved super-human level at playing Atari games [11] and Go [12], as well as impressive performance in practical problems such as autonomous driving [13]. These methods can inherently represent high dimensional state and continuous action-spaces. There are several works that model conflict resolution as a RL problem. Pham et al. [14] propose a similar approach. However, they do not consider a multi-agent environment and have a less specialized reward function. Ribeiro et al. [15] consider a single agent approach to conflict resolution through RL for unmanned aerial vehicles (UAVs). However, they do not use their model to issue maneuvers, but to enhance a current resolution algorithm. In this work, we attempt to overcome these drawbacks by modelling the resolution of pairwise conflicts as a multi-agent reinforcement learning (MARL) problem. We use Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [16] to train two agents, representing each aircraft in a conflict pair, that are capable of efficiently solving conflicts in the presence of surrounding traffic by considering heading and speed changes. We formalize the underlying Markov Decision process (MDP) by proposing a novel state representation which contains information such as position, heading and speed. Furthermore, we propose a highly specialised reward function that encourages efficient solutions and discourages solutions that are too conservative by considering several factors, such as time until loss of separation, closest point of approach, fuel consumption and surrounding traffic. The designed reward function can serve as a template on how to include the interests of different stakeholders in a resolution. The agents are trained and tested on real traffic, with a data augmentation technique to increase the variance of seen scenarios. Each scenario lasts 20 minutes, which is a common length for algorithms in the tactical level [8], [14]. Agents must take an action every 15 seconds, however our results show that agents are able to solve the majority of conflicts only in one time step. The agents are able to handle a continuous action space, with heading changes being capped at $\pm 45^o$ and speed changes at $[v - 6\%, v + 3\%]$ [10].

The rest of this paper is organized as follows: in Section II, we elaborate on the theoretical background necessary for this paper. In Section III, the experimental setup is presented. Results are presented and discussed in Section IV, while in Section V we draw conclusions and propose steps for further research.

II. Theoretical Background

A. Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of machine learning which deals sequential decision making [17]. In RL, an agent makes decision in an environment to optimize a certain notion of cumulative reward. The agent improves incrementally by modifying its behaviour according to previous experience. Furthermore, the RL agent does not require complete knowledge of the environment, it only needs to interact with it and gather information [18].

A given RL problem is usually formalized by a Markov Decision Process (MDP), which is a discrete time stochastic control process [19] that consists of a 4-tuple (S, A, T, R), where:

- S is the state space,
- A is the action space,
- $T : S \times A \times S \rightarrow [0, 1]$ is the transition function which is a set of conditional transition probabilities between states,
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward funtion

Practically, the agent starts at an initial state $s_0 \in S$. At each time step t, the agent has to take an action $a_t \in A$. Once this happens, the agent gets a reward $r_t \in R$ from the environment. The state transitions to $s_{t+1} \in S$. The agent stops interacting with the environment when it reaches a defined goal state. The agent's behaviour is encoded into a policy π which is a function that maps states to actions. Policies can be deterministic or stochastic.

B. Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is an extension of classical RL where there are more than one agents in the environment. This is formalized through partially observable Markov games [20], which are decision processes for N agents.

Similarly to MDPs, Markov games have a set of actions. However, in this case, the environment is not fully observable by the agents. Therefore, the Markov game has a set of observations $O_1, ...O_N$ for each agent. Every agent takes an action according to their policy and obtains a reward. Agents aim to maximize personal and total expected reward.

C. Multi-Agent Deep Deterministic Policy Gradient

Q-learning [17] is a popular method in RL that makes use of an action-value function for a policy π. It attempts to maximize the expected value of the total reward for a given and all successive steps. Q-learning algorithms that use neural networks for learning are called Deep Q-Networks (DQN) [11].

Policy Gradient methods are a group of methods that model and optimize the policy directly. The policy is modelled with a parametrized function with respect to parameters θ, $\pi_\theta(a|s)$. The goal of the methods is then to optimize the parameters θ for the best reward. Policy Gradient methods have been

found to outperform other methods in environments with stable dynamics [21].

In this work we will use *Multi-Agent Deep Deterministic Policy Gradient* (MADDPG) [16], which is an extension of single agent DDPG [22], where multiple agents must complete their tasks with only local information. For each agent, the environment is non-stationary as the policies of other agents are unknown. This leads to learned policies that only use individual observations of agents and no model of the dynamics of the environment.

MADDPG uses an actor-critic architecture, with agents and the critic being modelled as a neural network. The critic learns the value function (i.e. Q-learning), meaning that it is used to criticize the actions that are being taken. The network is updated from a Temporal Differences (TD) error. In MADDPG, the critic learns a centralized action-value function $Q^{\pi_i}(o_1, ..o_N, a_1, ..., a_N|\theta^Q)$ for an agent i. Each Q function is learned separately for all agents. This means that the critic is augmented with information about the policies of other agents.

The actor network $\pi_i(o_1, ..o_N, a_1, ..., a_N|\theta^\pi)$ learns the policy, meaning that it outputs an action in regard to its output. The actor only has access to local information and does not know the policies of other agents. Actors are encouraged to explore beyond their learned policies at each time step through Gaussian noise, which means that at each time step each actor has a probability of not following its policy but taking a random action. This step has been shown to improve the learned policies as actors can overfit their learned policies leading to worse overall performance [17], [20].

A known problem in MARL settings is the high variance caused by the interaction between agents present in the environment. MADDPG solves this by introducing policy ensembles. A collection of different sub-policies are trained for each agent. For every training episode, one particular sub-policy is randomly selected. Finally, the gradient update is done by taking all these sub-policies.

DQN methods, of which MADDPG is part of, often suffer from sample autocorrelation, leading to unstable training. Experience replay is used to mitigate such an issue [11]. In this technique each agent's experience is stored at each time step in a replay buffer. The memory is sampled randomly and is used to update the actor and critic networks. When the replay buffer becomes full, the oldest samples are discarded.

III. Experimental Setup

A. Data and Parameters Used

The model is evaluated using traffic data from Eurocontrol's DDR II, from 12.02.2019 [23]. Conflicts were detected using a simple state based method [24]. The trajectories of the present aircraft were projected in the future using a lookahead time of 300s, assuming constant speed and heading. If at any point during this time a horizontal and vertical separation infringement occurred (5 NM, 1000 feet at tactical level) the pair of aircraft were considered in a conflict. A filter was used to discard conflicts below FL250. Furthermore, except the conflict pair, we also keep several surrounding aircraft. We utilize the method proposed in [25], [26] to identify relevant aircraft. This procedure resulted in a total of 188 conflict scenarios.

As machine learning methods in general are sensitive to the data they are trained on, we employ a data augmentation technique to synthetically increase the variance of conflicts the model is trained on. For each scenario, we remove one of the conflict aircraft and create another one in conflict that has a different intrusion angle, closest point of approach (CPA) and time of separation loss than the removed conflict aircraft, while keeping surrounding non-conflict traffic. The values for the intrusion angle are in $[0, 30, 45, 60, 90]$, while those for CPA and time of separation loss are in $[1, 2, 4]$ NM and $[60, 120, 300, 600, 1200]$ seconds, respectively. This augmentation method is applied to each scenario and each conflict aircraft. The agents can only observe the 5 closest aircraft (including ownship).

The resulting scenarios are then divided into training and test sets with a ratio of 80%/20%. Training and test scenarios are kept apart in order to test the model in scenarios that it has never seen before.

Scenarios ran for 20 minutes and the agents had to make decisions every 15 seconds. The agent stops being active, i.e. does not make decisions anymore, if the conflict was resolved and it was back on track, or the scenario lasted longer than 20 minutes.

B. State Representation

One of the most important factors that can impact the learning capabilities and eventual performance of agents is how states in the given environment are formalized. Usually, the state is represented through a vector of a certain dimensionality, which should provide necessary information about the environment in order to facilitate the agents to be successful in their task. However, while providing more information in the state might result in an overall better performance [15], it is also followed by an increase in computational effort required to train a performant model.

Furthermore, the environment is highly non-stationary, as both conflict aircraft will change their policies in order to solve the conflict. This adds another layer of complexity to the representation of the state, which will need to be considered.

This work, represents the first steps to applying MARL to conflict resolution, therefore we opt for a more simple representation of the state. More specifically, the state is formalized through each present aircraft's position information. As we are only dealing with horizontal resolutions, we only use horizontal positions, i.e. latitude and longitude, heading and speed. This representation is shown in Figure 2, where 5 aircraft are present. Given a conflict pair present in the scenario, each conflict aircraft will observe the state information of the remaining aircraft.

C. The Reward Function

There is a wealth of research that outlines the importance of a suitable reward function in RL, especially applied to practical problems [16], [21], [27]–[30]. Ribeiro et al. [15] and Pham et al. [14] use a reward function solely based on the number of conflicts and number of losses of separation

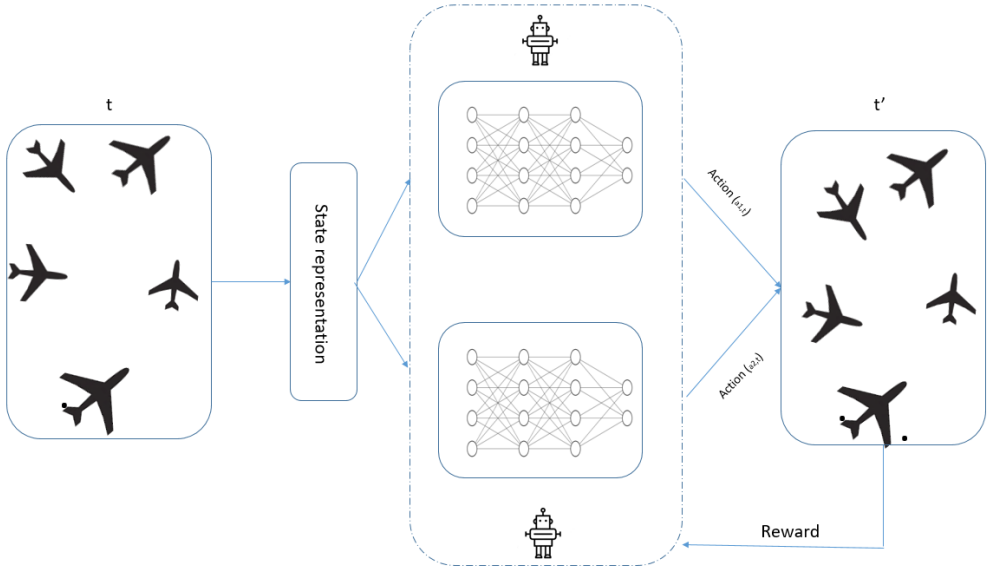

Figure 1: Model for conflict resolution.

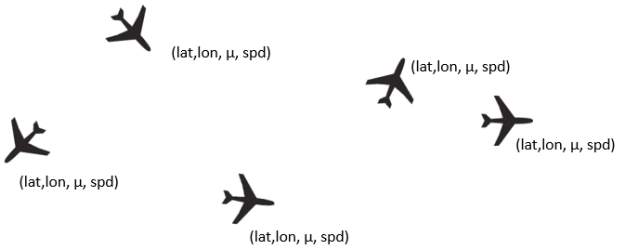

Figure 2: Position information of each aircraft in a scenario required to represent the state.

present in the scenario. However, not only are we concerned with solving the conflicts, we want to solve them as efficiently as possible. Thus, we take several factors into consideration to build the reward function:

1) Time until loss of separation and CPA: The model is encouraged to solve conflicts as soon as possible, in order to avoid dangerous situations. time until loss of separation and closest point of approach (CPA) are used to penalize the agents from slow solutions of conflicts with smallers CPAs. However, if, for example, the conflicting aircraft are almost in parallel, the agents will be penalized less if they take longer to solve the conflict.

2) Difference from track and optimal speed: To solve conflicts through minimal maneuvers, the agents are penalized for making big heading and speed changes. In order to achieve that, we give the aircraft negative reward the further they are from their track. Furthermore, as we assume aircraft to fly at their optimal speed, we penalize agents for deviating too much from it.

3) Fuel consumption: In order to discourage the model from taking actions that lead to big fuel consumption. In this work, we use the aircraft performance model OpenAP [31]. As per this model, the aircraft receives a negative reward for the amount of fuel it consumes each time step.

4) New conflicts: An undesirable behaviour of CDR algorithms is the inducing of new conflicts as a side effect of the resolution. Therefore, if the resolution proposed by the model induces a new conflict, for the given lookahead time, it is severely punished.

5) Airspace complexity: Airspace complexity is usually not an aspect that CDR algorithms take into consideration. However, complexity accounts for a large majority of ATCo workload [32], [33]. While we implicitly consider complexity by measuring if the resolution causes new conflicts, this alone is not informative enough in terms of complexity [32]. In this work, we consider the complexity formulation of Koca et al. [26]. They propose a hierarchical ecosystem structure, where relevant aircraft to the conflict are determined based on spatio-temporal interdependencies. The ecosystem is constructed as follows: the two aircraft of the conflict pair are the first order of the ecosystem. Aircraft that are affected by the conflict resolution are considered of the second order. Aircraft that are affected from the movement of the aircraft of the second order are considered of the third order and so on. To turn the information of the ecosystem into a single score, we do a weighted sum of the number of interdependencies for each order. This means that interdependencies get less and less important the further down the orders. In this way, we discourage resolutions that leave the airspace in a more complex situation.

All rewards are negative, as positive rewards can lead to the agent simply attempting at collecting as much reward and solving the conflict in an inefficient way [15]. The final reward is a weighted sum of all the single factors mentioned above.

D. The Model

In this work, we train two agents that represent each aircraft of the conflict pair. Figure 1 is a visual representation of the model. Each scenario consists of a conflict which needs to be resolved. When the conflict is detected, agents are randomly

assigned to the conflict aircraft. The agents then must attempt to resolve the conflict by maximizing their individual and global rewards (accumulative reward of single agents). At each time step while the conflict is not solved, each agent takes an action that is a combination of a heading and speed change. At the next time step, the agents receive a reward on how well the actions they took is perceived from the environment. The agents gain experience after each scenario they encounter and we ensure that agents have roughly the same experience by having only one initial conflict in the scenario.

The actor networks of each agent dictate what actions they have to take at every step. As only horizontal resolutions are considered in this work, the actor network outputs three values in the range $[-1, 1]$ (i.e. we apply the tanh activation function to the output layer), where two outputs are the sin and cos of the heading change angle α. The angle is then $tan^{-}1(\alpha) = sin(\alpha)/cos(\alpha)$. We put a maximum heading change of $\pm 45°$ per time step. The other output value is used to determine the new speed of the aircraft. In this work, we consider en-route traffic, therefore we assume that aircraft are initially flying at optimal speeds. As a result we limit speed change in an interval $[v - 6\%, v + 3\%]$ [10], [34].

As mentioned previously, the critic network is learned jointly for both agents. Furthermore, given that agents have the same reward structure, we can assume agents to be cooperative. However, no communication between agents is considered, which means that the only way agents are aware of the other agents policy is through the critic network.

E. Simulation Environment

Simulations were run on the Air Traffic Simulator BlueSky [35]. The simulator was chosen primarily because it is an open source tool, which allows for more transparency in the development and evaluation of the model proposed in this work. Furthermore, BlueSky has an Airborne Separation Assurance System (ASAS), which can support different CD&R methods. This allows for different resolution algorithms to be evaluated under the same conditions and scenarios.

IV. Simulation Results

A. Conflict Resolution

The algorithm was trained on the Google Cloud Platform[1] using an NVIDIA Tesla K80 GPU. The algorithm was then tested on 195 conflict scenarios with intrusion angles ranging from 0^o to around 140^o. Furthermore, scenarios had different CPAs, and time until the conflict started.

The model shows great promise with around 93% of conflicts solved using real traffic and considering as objective function a non-linear combination of reward factors. This means that the vast majority of conflicts are solved with both agents being involved in the solution and having no knowledge of the dynamics of the environment. Nevertheless, with such a big success rate, it is more informative to analyse different aspects of the resolutions such as, number of steps needed to solve the conflicts, maneuvers taken etc.

[1]https://cloud.google.com

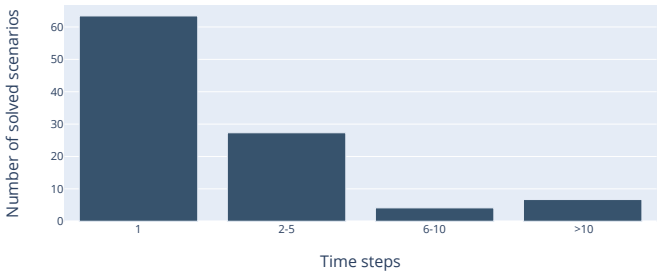

Figure 3: Number of steps required to solve the conflict. A time step is 15 seconds long.

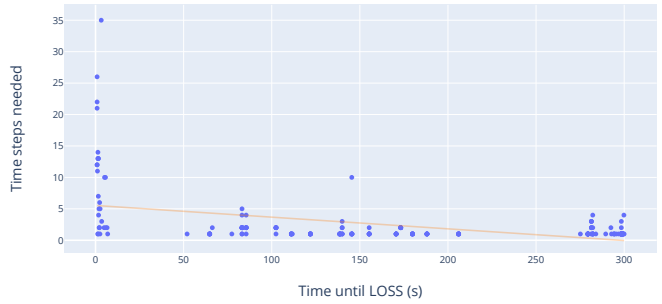

Figure 4: Relation between number of steps needed to solve the conflict and time before conflict starts.

As previously mentioned, the agents must take an action at every time step they are active, which means that at each time step the conflict aircraft must make a heading and speed change. Figure 3 shows the number of steps required to solve the conflicts. Scenarios where there were between 2 and 5, and 6 and 10 steps needed are grouped together, while scenarios that needed 1 step and more than 10 steps were shown separately. As shown in the figure, 63% of solved conflicts needed only 1 time steps. This is a promising result, which shows the model can learn by itself that the preferred behavior is to solve conflicts in one go.

Furthermore, we observe that conflicts that needed between 2 and 5 time steps to be solved represent around 27% of all solved conflicts. The other two groups represent less than 10% each. While these behaviors are not preferred, it is encouraging to see that the majority of conflicts are solved in less than 5 time steps. This means that the majority of conflicts are solved in around 1 minute from when the conflict was detected.

Figure 4 shows the relation between the number of time steps needed to solve the conflict and the time before the loss of separation (LOSS) occurred. Furthermore, the Pearson correlation coefficient between the two is calculated and resulted to be -0.43. This results indicates moderate negative correlation, which can also be seen from the figure. However, from the figure it is clear to see that the majority of conflicts that took multiple time steps to solve had a very low time

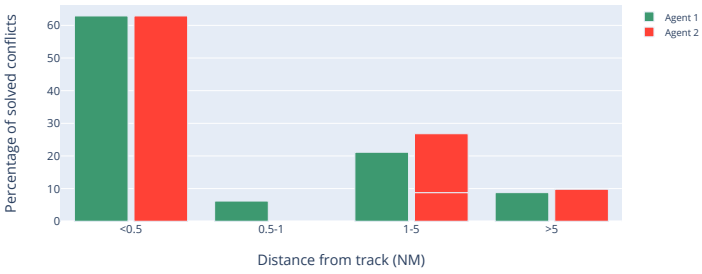

Figure 5: Distance of each agent from track in nautical miles when the conflict is resolved.

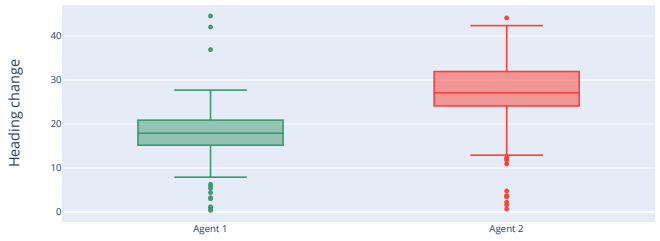

Figure 6: Average heading change needed to solve each conflict.

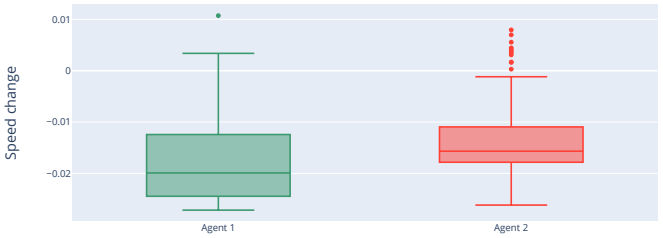

Figure 7: Average speed change needed to solve each conflict.

until LOSS. This means that for the majority of the time, the agents were trying to solve a LOSS. This is a surprising result, as one would expect that given the formulation of the reward function, the agents would react more when a LOSS is imminent. However, the agents have no model of the dynamics of the environment, except for the reward they receive. As a result, the big negative reward the agents get when the LOSS starts is not informative to the agents, and does not help them solve the conflict. To further support this claim, the agents were tested on 15 different scenarios, where the LOSS had already occurred when the agents were active. Out of those, they failed to solve the conflict 12 times. These results show that when given informative rewards from the environment, the agents are generally successful in solving the conflicts.

Figure 5 shows the distance from track at the end of the conflict for each agent. As one can see, both agents can solve around 65% of conflicts within 0.5 NM of their original track and around 70% within 1 NM. This result is promising, as resolutions that minimally displace the aircraft from their track are preferred. Both agents show a similar performance, which indicates the resolution would be likely acceptable by both aircraft. However, in this paper we assume cooperative agents, which can not always be expected in practice. The Pearson correlation coefficient between number of time steps required to solve the conflict and final distance from track is 0.78, which indicates high correlation. This shows that agents will increasingly deviate from track the longer they are not able to solve the conflict. While they eventually manage to solve the conflict, this resolution can not be accepted in practice, as it would require a huge deviation, which will result in major delays.

B. Actions Taken to Solve the Conflicts

In this section, we visualize and and discuss the actions taken by the agents to solve the conflicts. As mentioned previously, at each time step the agent could make a heading change of at most $\pm 45^o$ and a speed change in the range $[v - 6\%, v + 3\%]$.

Figure 6 shows as a box plot the heading change made by each agent to solve each conflict. For conflicts that required more than one time step to be solved, the average of all time steps is shown. Figure 7 shows the same information for the speed changes, with changes being the difference in percentage to the flight speed of the time step (which is assumed to be optimal). It is interesting to note that speed changes are almost 0, with each agent having an average decrease of -0.01%. This is a relevant result, as speed changes in this time frame are known to not be as efficient [8] and are not able to solve all conflicts. Furthermore, the penalization for bigger heading or speeding changes for the agents were of the same magnitude, meaning that no preference of actions was induced to the agents. As such, this result shows the agents' ability to learn desirable behavior with no previous knowledge of the environment. We also note that the majority of changes decrease the speed, which can be an indication from the performance model that the optimal speed can be improved.

For heading changes, results show each Agent 1 makes an average change of around 20^o, while Agent 2 makes an average change of around 31^o. An interesting result is the fact that big heading changes are made rarely by agents, with the maximum change being taken only once by each agent. This further shows the effect of the reward function in the behavior of the agents, as each prefer small changes that solve the conflict quickly. However, we note that Agent 2 makes on average a bigger heading change. Nevertheless, the heading changes are still relatively small, which is preferred.

Furthermore, the fact that the majority of conflicts are solved only in one time step shows that the learned behavior

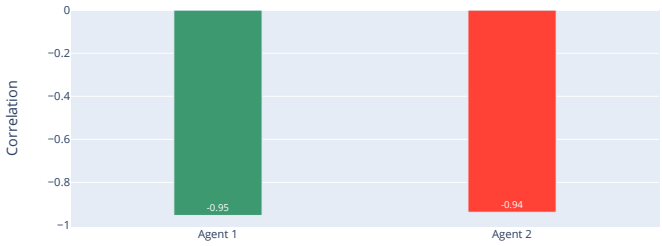

Figure 8: Correlation between final reward and time steps needed to solve the conflict for each agent.

Figure 9: Direction and magnitude of heading of each agent for 5 example scenarios.

of the agents can give resolutions that optimize several factors at once. Additionally, the agents were penalized heavily if the resolution they proposed would increase the complexity of the airspace, or even create a new conflict with one of the surrounding aircraft. We do not observe new conflicts that were created as a result of a resolution. This is further evidence of agents being able to learn positive behaviors, as ATCos do not issue maneuvers that create new conflicts.

C. Agent behavior

In this work, we assume cooperative agents. As such, both agents have the exact reward structure and furthermore, must maximize a global notion of reward.

Given the non-linear nature of neural networks, one cannot expect a linear correlation between the factors that take part in the reward function and the actions taken to solve the conflict. For instance, results indicate no correlation between the magnitude of heading change with the CPA and time until LOSS. Such a conclusion, however, speaks in favor of the model, as it shows that the agents are capable of extracting more complicated relations between reward factors rather than being highly influenced by one of them. Nevertheless, the effects of individual factors in the reward function can be observed through the results. For instance, in Figures 6 and 7, we see that the model learns to prefer heading changes as opposed to speed changes, which cannot solve all conflicts. Furthermore, given that in this work we assume that aircraft are flying at optimal speed, the strategy of small heading changes and not changing speed comes as a result of penalising high fuel consumption in the reward function. Figure 8 shows the correlation between final reward and number time of time steps needed to solve the conflict. The results indicate a clear negative correlation, which suggests that the longer it takes for the agents to solve the conflict, the more they will be penalized by the reward. While this is an expected result, it is an important one, as it further confirms the non-linear relation between reward factors that the neural networks induce. Furthermore, this result shows why the agents tend to solve conflicts so quickly. The behavior, which is influenced by the reward, coincides with usual controller behavior, as they usually issue one maneuver to solve the conflict.

Another interesting aspect of the heading changes is the direction of the change. To measure it, we compare the direction of the average heading change for all solved conflict scenarios. Results show that in 83% of the cases the agents solve the conflicts by going in the same direction, while in 16% of the cases they go in different directions. Figure 9 shows the heading changes made by each agent in 5 example scenarios. As it can be noted, Agent 2 in all cases makes a bigger change, even in cases where the aircraft go in different directions. This result is unexpected and does not reflect how conflicts are generally solved in practice. Nonetheless, this result explains the difference in average heading between Agent 1 and Agent 2 shown in Figure 6. In this case, Agent 2 has to make a bigger turn to solve the conflicts in the majority of cases.

In the setting of this paper, where we assume cooperative agents that do not communicate, this does not affect the effectiveness of the algorithm, as the ultimate goal is to solve the conflicts. Furthermore, the agents have no model on the dynamics of the environments, thus they have no concept of how conflicts are usually solved. Additionally, a known problem with MARL algorithms is the high non-stationarity of the environment. Each agent takes an action at each time step and they do not know the policy of the other agent. Thus, they observe the actions taken by the other agent as a constant change in the environment. As a result, injecting more expert knowledge into the state representation of the environment, it can be expected that these peculiarities in agents' behavior can be resolved. Furthermore, communication module between agents, in the form of knowledge of each other's policies can be expected to do the same.

Finally, an important result is the fact that both agents are actively attempting to solve the conflicts. In fact, in only 1% of the seen scenarios it happens that one of the agents makes a turn of less than 5^o and the other makes a turn of bigger than 5^o. This result shows that the agents successfully avoid "freeloading", which is undesirable behavior.

V. Conclusions and Future steps

A. Contributions

In this paper, we tackle the problem of conflict resolution at the tactical level in the presence of surrounding traffic by

modeling it as reinforcement learning problem. We utilize, MADDPG, which is a multi-agent reinforcement learning algorithm. It consists of a actor-critic architecture, where an actor corresponds to an agent taking actions and the critic models the Q-values. To the best of our knowledge, this is the first work that uses a multi-agent reinforcement learning approach to conflict resolution.

We propose a novel state representation consisting of position information, heading and current speed. Furthermore, we propose a reward function that not only optimizes for number of conflicts solved, but encourages efficient solutions. Factors that are included in the reward function are fuel consumption, CPA, time to LOSS, the creation of new conflicts and airspace complexity. The reward function evaluated in this work, can serve as a template for other research that goes in the same direction.

The model is trained and tested on conflict scenarios from real traffic, with a data augmentation technique applied to increase the variance of encountered conflict geometries. The scenarios last 20 minute and in each scenario, the conflict pair will be assigned an agent, which will take actions every 15 seconds. In this work, agents are able to handle continuous actions space, which means that we do not prescribe fixed maneuvers to solve the conflict. This overcomes a common limitation of exisitng research, where fixed maneuvers are usually issued. Each action consists of a heading ($\pm 45^o$) and speed change ($[v - 6\%, v + 3\%]$).

Results indicate an impressive resolution success rate of 93%. Furthermore, the agents are able to learn several desired behaviors, while having no model of the dynamics of the environment. First of all, the majority of conflicts are solved only in one time step, which emulates how conflicts are solved in practice. Furthermore, the majority of conflicts are solved with relatively small heading changes. This indicates that the proposed reward function directs the agents not only to solve the conflicts as soon as possible but as efficiently as possible. Further evidence to this is the fact that the speed changes the agents make are negligible, with the average being -0.01%. This is an interesting result, as speed changes are generally considered to be less efficient and are not able to solve all conflicts.

B. Challenges and Future Steps

Nevertheless, there are several challenges to be considered to our initial approach. First of all, the agents are not able to solve all conflicts. Results are promising, but a safety critical machine learning approach should come with resolution guarantees. Furthermore, there are cases where agents behave in peculiar ways. For instance, in the majority of cases where they have little time before LOSS, the agents are not able to solve the conflicts. This is somewhat counterintuitive, as one would expect the agents to make a bigger heading change to solve the conflict. A possible explanation to this can be the calibration of the reward function. Further investigation to the failed cases indicates that in the aforementioned scenarios, the agents get penalized too much. As a result, the other factors in the reward function are unable to guide the agents out of the conflict.

Additionally, in most cases, agents make their heading changes in the same direction. To this effect, Agent 1 makes on average a turn of 20^o while Agents 2 makes a turn of 30^o, This is not a natural way of solving conflicts. Furthermore, such resolutions might not be accepted by ATCos. One possible solution to this issue could be the inclusion of a more informative and expressive complexity metric. Such a metric should give more detailed and multi faceted complexity information. As a result, agents will be discouraged from taking actions that increases complexity. In addition, in this work, after the resolution of the conflict, aircraft are send back on their tracks with the angle opposite heading and speed change that they solved the conflict. An interesting addition to this approach would be to let the agents learn what the best way back could be. After resolution, positive reinforcement could be used to incentivize agents to quickly and safely go back on track.

In this work we assume cooperative agents. While this is a valid approach, it might not reflect the whole reality of the situation. In practice, aircraft might not be willing to make certain maneuvers. Another interesting approach would be to model different behaviors of the agents, such as competitive behavior. In such an approach, agents would not have the same reward structure and would have certain preferences towards certain resolution methods. This would make for a valuable comparison in terms of local and global reward optimization. Furthermore, one way to improve resolutions would be to take feedback from controllers. In such approaches, the agents get a reward not only from the environment but also from a teacher, which can help alleviate issues from the non-stationarity of the environment and eventually make convergence easier. In addition, the models are trained and tested on a specific dataset. This dataset is not representative of all possible conflicts scenarios and geometries and When the agents are presented with unseen conflict situations, they may fail to solve them. A solution to this is to introduce lifelong learning, which is an approach to machine learning that retrains itself when faced with unseen data.

Ultimately, a conflict resolution tool that is based on machine learning will still need to be monitored by ATCOs. Such methods need to have a high degree of explainability. More specifically, agents need to be able to show some reasoning on how they picked actions. Ways how these explanations can be informative in a meaningful way presents a very interesting research question.

ACKNOWLEDGMENT

This research has received funding from the SESAR Joint Undertaking under the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 783287. The opinions expressed herein reflect the authors' view only. Under no circumstances shall the SESAR Joint Undertaking be responsible for any use that may be made of the information contained herein.

REFERENCES

[1] Giulio Di Gravio, Maurizio Mancini, Riccardo Patriarca, and Francesco Costantino. Overall safety performance of the air traffic management

system: Indicators and analysis. *Journal of air transport management*, 44:65–69, 2015.

[2] Andrew Cook, Seddik Belkoura, and Massimiliano Zanin. Atm performance measurement in europe, the us and china. *Chinese Journal of Aeronautics*, 30(2):479–490, 2017.

[3] Arnab Majumdar and Washington Y Ochieng. Factors affecting air traffic controller workload: Multivariate analysis based on simulation modeling of controller workload. *Transportation Research Record*, 1788(1):58–69, 2002.

[4] Jun Tang. Conflict detection and resolution for civil aviation: A literature survey. *IEEE Aerospace and Electronic Systems Magazine*, 34(10):20–35, 2019.

[5] Brenda Carpenter, James Kuchar, and Carpenter. Probability-based collision alerting logic for closely-spaced parallel approach. In *35th Aerospace sciences meeting and exhibit*, page 222, 1997.

[6] Lucia Pallottino, Eric M Feron, and Antonio Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE transactions on intelligent transportation systems*, 3(1):3–11, 2002.

[7] Jacco M Hoekstra, Ronald NHW van Gent, and Rob CJ Ruigrok. Designing for safety: the 'free flight'air traffic management concept. *Reliability Engineering & System Safety*, 75(2):215–232, 2002.

[8] Clément Peyronne, Andrew R Conn, Marcel Mongeau, and Daniel Delahaye. Solving air traffic conflict problems via local continuous optimization. *European Journal of Operational Research*, 241(2):502–512, 2015.

[9] Yang Yang, Jun Zhang, Kai-Quan Cai, and Maria Prandini. Multi-aircraft conflict detection and resolution based on probabilistic reach sets. *IEEE Transactions on Control Systems Technology*, 25(1):309–316, 2016.

[10] Romaric Breil, Daniel Delahaye, Laurent Lapasset, and Éric Féron. Multi-agent systems for air traffic conflicts resolution by local speed regulation and departure delay. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2016.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[12] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[13] Maria Huegle, Gabriel Kalweit, Branka Mirchevska, Moritz Werling, and Joschka Boedecker. Dynamic input for deep reinforcement learning in autonomous driving. *arXiv preprint arXiv:1907.10994*, 2019.

[14] Duc-Thinh Pham, Ngoc Phu Tran, Sameer Alam, Vu Duong, and Daniel Delahaye. A machine learning approach for conflict resolution in dense traffic scenarios with uncertainties. In *ATM Seminar 2019, 13th USA/Europe ATM R&D Seminar*, 2019.

[15] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. Determining optimal conflict avoidance manoeuvres at high densities with reinforcement learning. *10th SESAR Innovation Days*, 2020.

[16] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[17] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[18] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

[19] Richard Bellman. Dynamic programming and stochastic control processes. *Information and control*, 1(3):228–239, 1958.

[20] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[21] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[23] Thimjo Koca, Ralvi Isufaj, and Miquel Angel Piera. Strategies to mitigate tight spatial bounds between conflicts in dense traffic situations.

[24] James K Kuchar and Lee C Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on intelligent transportation systems*, 1(4):179–189, 2000.

[25] Marko Radanovic, Miquel Angel Piera Eroles, and Thimjo Koca. Adaptive aerial ecosystem framework to support the tactical conflict resolution process. 2017.

[26] Thimjo Koca, Miquel Angel Piera, and Marko Radanovic. A methodology to perform air traffic complexity analysis based on spatio-temporal regions constructed around aircraft conflicts. *IEEE Access*, 7:104528–104541, 2019.

[27] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using trajectory data to improve bayesian optimization for reinforcement learning. *The Journal of Machine Learning Research*, 15(1):253–282, 2014.

[28] Nina Schefers, Juan José Ramos González, Pau Folch, and José Luis Munoz-Gamarra. A constraint programming model with time uncertainty for cooperative flight departures. *Transportation Research Part C: Emerging Technologies*, 96:170–191, 2018.

[29] JC Van Rooijen, I Grondman, and R Babuška. Learning rate free reinforcement learning for real-time motion control using a value-gradient based policy. *Mechatronics*, 24(8):966–974, 2014.

[30] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

[31] Junzi Sun, Jacco M Hoekstra, and Joost Ellerbroek. Openap: An open-source aircraft performance model for air transportation studies and simulations. *Aerospace*, 7(8):104, 2020.

[32] Parimal Kopardekar, Jessica Rhodes, Albert Schwartz, Sherri Magyarits, and Ben Willems. Relationship of maximum manageable air traffic control complexity and sector capacity. In *26th International Congress of the Aeronautical Sciences (ICAS 2008), and AIAA-ATIO-2008-8885, Anchorage, Alaska, Sept*, pages 15–19, 2008.

[33] Gano Chatterji and Banavar Sridhar. Measures for air traffic controller workload prediction. In *1st AIAA, Aircraft, Technology Integration, and Operations Forum*, page 5242, 2001.

[34] Romaric Breil, Daniel Delahaye, Laurent Lapasset, and Eric Féron. Multi-agent systems to help managing air traffic structure. *Journal of Aerospace Operations*, 5(1-2):119–148, 2017.

[35] Jacco M Hoekstra and Joost Ellerbroek. Bluesky atc simulator project: an open data and open source approach. In *Proceedings of the 7th International Conference on Research in Air Transportation*, volume 131, page 132. FAA/Eurocontrol USA/Europe, 2016.

AUTHOR BIOGRAPHIES

Ralvi Isufaj received his bachelor's degree in computer engineering at the Polytechnic University of Tirana, Tirana in Albania in 2014 and his master's degree in 2018 in computer science with a specialization in artificial intelligence at the University of Freiburg, Freiburg, Germany. Since 2019, he is pursuing a Ph.D. degree with the Department of Telecommunications and System Engineering, Autonomous University of Barcelona, Barcelona, Spain. His research line is on Machine Learning applications to air conflict resolution

David Aranega Sebastia is pursuing a master degree in mathematical modelling in the Autonomous University of Barcelona. He graduated as a mechanical engineer in 2019 from Universitat de Lleida. His interests lie in modeling techniques and artificial intelligence technologies

Miquel Angel Piera received the degree (Hons.) in computer engineering from the Autonomous University of Barcelona, Barcelona, Spain, in 1988, the M.Sc. degree in control engineering from the University of Manchester, Institute of Science and Technology, Manchester, England, 1991, and the Ph.D. degree from the Autonomous University of Barcelona, in 1993. He is a delegate for Technical Innovation Cluster with the Autonomous University of Barcelona, where he is currently a full-time Professor with the System Engineering Department.