EXPLORATORY RESEARCH

sesar JOINT UNDERTAKING

D5.22 PhD final report: Deep Multi-Agent Reinforcement Learning Applications in ATM

Deliverable ID:	**5.22**
Dissemination Level:	**PU**
Project Acronym:	**Engage**
Grant:	**783287**
Call:	**H2020-SESAR-2016-2**
Topic:	**SESAR-ER3-01-2016 Knowledge Transfer Network**
Consortium Coordinator:	**University of Westminster**
Edition date:	**23 August 2022**
Edition:	**01.00.00**
Template Edition:	**02.00.05**

EUROPEAN PARTNERSHIP

Co-funded by the European Union

Engage

THE SESAR KNOWLEDGE TRANSFER NETWORK

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 783287 under European Union's Horizon 2020 research and innovation programme.

Abstract

This is the final report of the *Deep Multi-Agent Reinforcement Learning Applications in ATM* PhD, which was awarded funding through the Engage KTN's Call for PhDs and post-graduate theses. This report provides a summary of the research in advance of the published PhD thesis, which can be accessed directly from the Universitat Autònoma de Barcelona (the link is provided in Section 11.1).

SESAR Engage KTN – PhD final report

PhD title:	Deep Multi-Agent Reinforcement Learning Applications in ATM
Candidate's name:	Ralvi Isufaj
Lead supervisor's name:	Miquel Angel Piera
Co-supervisor's name (if applicable):	-
Proponent institute:	Autonomous University of Barcelona
Consortium institutes (if applicable):	-
Thematic challenge:	TC2 Data-driven trajectory prediction
Edition date:	19 August 2022
Edition:	1.1
Dissemination level:	Public

This project has received funding from the SESAR Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 783287.

1. Abstract

Air Traffic Management's (ATM) aim is to ensure separation management of aircraft in an efficient way, minimizing possible delays and costs. The expected increase in air traffic demand across manned and unmanned traffic requires a higher level of automation to support the decision making. Adaptive self-Governed aerial Ecosystem by Negotiated Traffic (AGENT) was an exploratory research project supported by the H2020 Research and Innovation Programme, which proposed a system where the avoidance of potential loss of separations is done in a distributed and collaborative way while the controllers monitor the process. This PhD project is built on AGENT's future work proposals and seeks possible improvement of several critical aspects of the system through the application of Machine Learning (ML) techniques. There were two clear goals in this project: define airspace complexity in a way that challenges current definitions and overcomes their limitations and investigate how ML can be applied to safety in aviation. We investigate these problems in en-route traffic at the tactical level, as well as UAV systems.

The first major contribution of this thesis has been modelling air traffic as a graph in the context of airspace complexity and conflict resolution. We define a graph with aircraft as nodes and interdependencies between them as edges of the graph. This definition allows for problem specific definitions of interdependencies. We further extend the definition of air traffic as a graph by including the time domain, which creates dynamic graphs. We define airspace complexity as graph connectivity and propose four indicators that combine different topological information and the severity of interdependencies to give a complete and nuanced picture of complexity. These indicators are able to provide a dynamic evolution of complexity by leveraging the modelling choice of air traffic as a dynamic graph. Simulation results indicated that the indicators we propose give detailed information and overcome drawbacks of existing metrics. We evaluated our approach using real and synthetic traffic and demonstrated that the indicators express different facets of complexity, confirming that all indicators are needed. The way we define complexity also provides a new framework in the design of conflict resolution algorithms which considers the reduction of airspace complexity in addition to safety preservations. Conflict Resolution (CR) algorithms could be discouraged from providing solutions that increase the overall complexity of the airspace.

Furthermore, we model CR as Multiagent Reinforcement Learning Problem (MARL). We initially investigate CR only in a pairwise setting using Multiagent Deep Deterministic Policy Gradient (MADDPG) as a learning algorithm. We propose a novel state representation that combines positional information with speed and heading of the aircraft. Additionally, we propose a reward function that not only guides agents towards solving the conflict but also to consider factors such as fuel consumption, airspace complexity and delays. Our results indicate that the agents are capable of solving the conflicts and further learning desired behaviours such as solving them as soon as possible with minimal manoeuvres. However, this method suffers from issues of scalability and nonstationarity. In order to overcome these issues, we utilize Graph Neural Networks (GNNs). GNNs inherently allow communication between agents which facilitates cooperation between them. We apply Graph Convolutional Reinforcement Learning (DGN) in CR for Unmanned Aerial Vehicles (UAV) to solve conflicts with 3 and 4 present aircraft which we assume to be cooperative. We achieve impressive performance with the agents being able to always solve the conflicts. Furthermore, they learn a strategy that increases the distance between them, without previous knowledge of the environment. Currently, we are using this application domain to investigate some fundamental questions in MARL such as agent coordination, heterogeneity and transparency in environments where agents have individual and common goals.

2. Objective of the study

The main objective of this PhD project was to investigate how the ability of agents to learn can affect the performance of CR algorithms. More specifically, if air traffic is to be modelled as a multiagent system, are they able to learn how to resolve conflicts in a purely data driven fashion? We model CR as a MARL problem with agents having no previous knowledge of the environments and with no expert knowledge being imposed on them.

In order to make CR algorithms that not only solve conflicts but also consider a wider view of the airspace, we sought to define airspace complexity in a way that challenges current definitions and overcomes their limitations. We investigate these problems in en-route traffic at the tactical level, as well as UAV systems.

3. Motivation

The mission of air traffic management (ATM) is to make air traffic possible by means of efficient, environmentally friendly and socially valuable systems. In the current situation, challenges such as sustainability, the environmental impact and fuel consumption 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 resolution algorithms have been a prominent research within the ATM community, with many models being proposed. However, existing methods mainly deal with pairwise conflicts. This however, is not an assumption that might hold for long, with the expected increase in (manned or unmanned) air traffic density, where encounters with more than two aircraft are likely to happen.

4. Advances this work has provided with regard to the state of the art

In this section we will explicitly list out the contributions of this PhD project:

- Air traffic as a graph – To the best of our knowledge, our work in this thesis is one of the few that extensively makes use of graph theory in the study of air traffic and the first to do so in the context of airspace complexity and CR. We model air traffic as a dynamic graph (i.e., changes over time) and this allows us to employ the wealth of research conducted on graphs. Furthermore, our definition is flexible in such a way that other researchers can make problem specific modifications to adapt this definition to their needs. In [23], the authors also model air traffic as a graph to tackle demand-capacity imbalances. However, their work has a different focus and an unrelated method for defining interdependencies between aircraft.
- Airspace complexity – We define four complexity indicators based on graph theory that attempt to overcome challenges of current complexity definitions. First of all, we provide a dynamic evolution of complexity and do not limit it to a single score. In such a way, the

complexity indicators can also be used to guide strategical, pre-tactical and tactical actions for a smooth flow of aircraft. The indicators combine spatiotemporal topological information to give a complete and nuanced picture of complexity.

- Conflict Resolution as a Multiagent Reinforcement Learning problem – In this thesis, we provide a full description of how to model CR as a MARL problem starting from the formalisation of the MDP to scaling up to multiple agents. Furthermore, we give examples of how a reward function that takes into consideration multiple factors in addition to solving conflicts could look like. In addition, we have shown that GNNs can be successfully applied to air traffic in such a way that facilitates communication and cooperation between agents.

5. Methodology

The work of this thesis relies largely on concepts from graph theory and reinforcement learning. Throughout the thesis there has been overlap between these two methodologies, however for the sake of clarity they will be elaborated separately.

Graph Theory

Many real-world situations can be described by a diagram consisting of a set of points together with lines joining certain pairs of these points. For instance, these pairs could represent people in a social network, with lines joining friends; or the points could be aircraft with lines representing interdependencies between them. *Graphs* are the mathematical objects that model these kinds of pairwise relations.

An **undirected** graph G = (V, E) is a mathematical structure that consists of a set V of elements called vertices and a set E of pairs of vertices called edges. Let e = (a,b) be an edge of V. Then e joins two vertices a, b in V and is called incident of a and b. In turn, those vertices are called the endpoints of e and they are adjacent to each other.

The **degree** of a vertex of a graph is the number of edges that are incident to the vertex. The **order** of a graph G = (V, E) is $|V|$, while the **size** of the graph is $|E|$.

A **triplet** is a group of three vertices that are fully connected, i.e., any pair of the three vertices are connected by an edge. We denote with T the set of all triplets in the graph.

H = (U, F) is a **subgraph** of G if the vertices and edges of H are subsets of the vertices and edges of G, i.e. $U \subseteq V$ and $F \subseteq E$.

Similar to undirected graphs, **directed** graphs (digraphs) can also be defined. The difference in definition is that in the case of directed graphs, E is now a set of ordered edges. All attributes can be adapted. Graphs can be **weighted** or **unweighted.** In the case of weighted graphs, each edge is assigned a number (the weight) for example between 0 and 1. If not edge connects two vertices, the weight is 0. An important attribute of weighted graphs is the **strength** of a vertex. Its definition is analogous to the degree, but it takes into consideration the weights:

$$s(i) = \sum_{j=1}^{N} w_{i,j}$$

In addition to the visual representation of a graph, there are several ways a graph can be described. The most common way is the **adjacency matrix** A, which for unweighted graphs is:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} a_{i,j} = \begin{cases} 1, \text{ if } (v_i, v_j) \in E \\ 0, \text{ otherwise} \end{cases}$$

The idea behind this representation is this: build a matrix with all possible edges between all vertex pairs of a graph. If an edge is actually present in the graph, then the entry in the matrix is 1, otherwise it is 0. Similarly, a weighted graph can also be represented with an adjacency matrix:

$$A_w = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} a_{i,j} = \begin{cases} w_{i,j}, \text{ if } (v_i, v_j) \in E \\ 0, \text{ otherwise} \end{cases}$$

In the weighted case, the entries for the edges present in the graph correspond to weight of the edge. In the case if an undirected graph, the adjacency matrix is symmetric, while for directed graphs this does not hold.

Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of machine learning which deals with sequential decision making [1]. A given RL problem is formalized by a Markov Decision Process (MDP), which is a discrete time stochastic control process [2] that consists of a 4-tuple (S, A, T, R), where:

- S is the state space,
- A is the action space,
- T: S x A x S -> [0,1] is the transition function which is a set of conditional probabilities between states,
- R: S x A -> **R** is the reward function

In RL, an agent makes decisions in an environment to maximize a certain notion of cumulative reward G, defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Where γ is a discount factor between 0 and 1. Its task is to inform the agent how relevant immediate rewards are in relation to rewards further in the future. The higher γ is the more the agent will care about future consequences.

The agent improves incrementally by modifying its behaviour according to previous experience. The agent does not strictly require complete information or knowledge of the environment; it only needs to interact with it and gather information [3].

The RL agent starts at an initial state $s_0 \in S$ and at each time step t must take an action $a_t \in T$. Then, the agent gets a reward $r_t \in R$ from the environment. The state then transitions to the next state which is dictated by the taken action and the dynamics of the environment. Finally, 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 can be deterministic π: S -> A, or stochastic π: S x A -> [0,1]. The policy uses the reward function implicitly, meaning that in the best case, the policy will guide the agent to the states with the most reward.

There are two ways that are used to predict the total future discounted reward: the value function V and the action-value function Q, defined as follows:

$$V^{\pi}(s) = \mathbb{E}_{\pi}(R_t|s_t = s)$$
$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}(R_t|s_t = s, a_t) = a$$

The value function represents the future expected reward in the current state if the policy is followed, while action-value function represents expected rewards for state-action pairs following policy π. Ultimately, the goal of all RL algorithms is to solve either of these functions.

Q-learning is one most prominent algorithms for solving RL problems. There, an agent must learn to estimate the optimal action-value function in the form of a table with as many state-action pair entries as possible [4]. However, in cases where the state space or action space (or both) are continuous, there are infinitely many state-action pairs, which makes it unfeasible to store the values in table. In those cases, a function is used to approximate the Q function. Such a function with parameters μ is optimized through an objective function based on the Bellman equation [2].

In the case of Deep Q-Networks (DQN) [5], the Q function approximators are neural networks. However, several issues arise when applying deep learning directly on a RL problem. First, in RL rewards can be sparse or delayed, which hinders neural networks, as they rely on directly gained feedback. Additionally, the data that are obtained from an RL problem are highly correlated and lastly, the data distribution changes as the policy does, making it nonstationary, which further impairs the learning capabilities of neural networks. To overcome these issues, several modifications must be made. Experience replay is used to mitigate the issue of sample autocorrelation [5]. In this technique, the agent's experience is stored at each time step in a replay buffer. The memory is sampled randomly and is used to update the networks. When the replay buffer becomes full, the simplest solution is to discard the oldest samples. The nonstationarity of the data makes the training unstable, which can lead to undesired phenomena such as *catastrophic forgetting*? Where the agent suddenly ``forgets'' how to solve the task after apparently having learned a suitable policy. Such an issue can be mitigated using *target networks*, which is an identical network to the one used to learn the Q function, that is held constant to serve as a stable target for learning for a fixed number of time steps.

Multiagent Reinforcement Learning

Multiagent 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, 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_2…O_N for each agent. Similarly, to single agent RL, in the MARL setting, agents take actions according to their policy and obtain rewards. The goal of the agents is to maximize personal and total expected reward.

6. Description of the data the study relies on

For all of the work presented in this report, we utilized BlueSky [6] as the simulator, and overall training and testing environment for all our experiments. This simulator was chosen primarily because it is an open source tool, allowing for more transparency in developing and evaluating the developed models. The modular nature of BlueSky allows for different resolution algorithms to be evaluated under the same conditions and scenarios. Finally, BlueSky has an Airborne Separation Assurance System (ASAS) that supports and allows for new CD&R methods.

In the first steps of the thesis, the goal was to use EUROCONTROL's DDR II data. We managed to conduct experiments using DDR data for the complexity indicators. For that paper, we utilized flown trajectories from 12.02.2019. As we were only interested in en-route traffic, a filter was used to discard traffic below FL250. Furthermore, we make use of sector data for the same day to make our evaluations as realistic as possible.

Some of this data was used to generate the dataset for the paper "Towards Conflict Resolution with Deep Multi-Agent Reinforcement Learning". However, in addition we augmented these trajectories with an algorithm we developed. This was done to synthetically increase the variance of conflicts the model is trained on. The algorithm, shown in Algorithm 1, takes as input a conflict pair and for each aircraft in the pair generates new conflict geometries with varying headings, Closest Point of Approach (CPA), time until loss etc. For each scenario, we remove one of the conflict aircraft and create another one in conflict that has a different intrusion angle (i.e., conflict 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. These values are based on the geometries we encountered from the original 188 scenarios with some values to test more extreme situations, such as a head-on conflict with a CPA of 1 NM. This augmentation method is applied to each scenario and each conflict aircraft. We note that the scenarios are not augmented with all possible permutations of these values, resulting in around 1000 total scenarios. This is done in order to maintain a reasonable training time for the model. 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.

Algorithm Data Augmentation Algorithm

procedure AUGMENT($scenario$)
- $scenario_{new} \leftarrow \emptyset$
- $hdgs_{conflict} \leftarrow [0, 30, 45, 60, 90]$
- $cpa \leftarrow [1, 2, 4]$
- $t_{conflict} \leftarrow [60, 120, 300, 600, 1200]$
- $ac_{conflict} \leftarrow IDENTIFY-CONFLICT(scenario)$
- $ac_{nonconflict} \leftarrow IDENTIFY-NONCONFLICT(scenario)$
- $ac_{chosen} \leftarrow RANDOM(ac_{conflict})$
- $hdg_{chosen} \leftarrow RANDOM(hdgs_{conflict})$
- $cpa_{chosen} \leftarrow RANDOM(cpa)$
- $t_{chosen} \leftarrow RANDOM(t_{conflict})$
- $ac_{new} \leftarrow CREATE-CONFLICT(ac_{chosen}, hdg_{chosen}, cpa_{chosen}, t_{chosen})$
- $scenario_{new} \leftarrow ac_{conflict} \cup ac_{new} \cup ac_{nonconflict}$
- **return** $scenario_{new}$

end procedure

Algorithm 1 Data Augmentation algorithm for "Towards Conflict Resolution with Deep Multi-Agent Reinforcement Learning"

For the remaining of the work in this thesis, we moved to fully synthetic traffic, as our application domain switched towards UAVs. We follow a similar logic as to the algorithm mentioned above to generate diverse conflict geometries in different airspaces. To create the multi-UAV conflict, first, a reference aircraft is initialized, with a heading sampled from a uniform distribution from 0◦ to 360◦. Then, this aircraft is added to the set of created aircraft. To generate the rest of the conflicting UAVs, we sample from the set of the created ones. Then, a conflict angle is chosen from the list [0◦, 45◦, 90◦, 90◦, 135◦, 180◦, −135◦, −45◦]. Next, to add some variance to the intrusion headings, a variance in the range [−10◦, 10◦] is added to each case. After that, the severity of the conflict is decided by sampling from a uniform distribution between 0.1 and 1. Finally, we set the time the new aircraft enters in conflict with the randomly chosen aircraft to 15 s. The CRECONF function is taken from the BlueSky simulator, and it provides the location and speed of a new conflicting aircraft. However, as compound conflicts have temporal boundaries, no accidental conflicts are added in one look-ahead time, which is set to 8 s. This is checked by the CONFLICT function, also taken from BlueSky.

Algorithm Data Generation Algorithm

procedure GENERATE($target, spd_{min}, spd_{max}, t_{loss}, t_{la}$)
  $created \leftarrow \emptyset$
  $hdgs_{conflict} \leftarrow [0, 45, 90, 135, 180, -45, -135, -90]$
  $var \leftarrow 10$
  $lat_{ref} \leftarrow 41.4$
  $lon_{ref} \leftarrow 2.15$
  $spd_{ref} \leftarrow$ UNIFORM($speed_{min}, speed_{max}$)
  $hdg_{ref} \leftarrow$ UNIFORM(1,360)
  $ac_{ref} \leftarrow$ AIRCRAFT($lat_{ref}, lon_{ref}, spd_{ref}, hdg_{ref}$)
  $created \leftarrow created \cup ac_{ref}$
  while SIZE($created$) $<$ $target$ **do**
    $accepted \leftarrow$ False
    while $accepted$ is False **do**
      $hdg \leftarrow$ SAMPLE($hdgs_{conflict}$)+UNIFORM(-var, var)
      $severity \leftarrow$ UNIFORM(0.1,1)
      $cpa \leftarrow threshold - (threshold \times severity)$
      $chosen \leftarrow$ SAMPLE($created$)
      $ac_{proposed} \leftarrow$ CRECONF($hdg, cpa, tloss, chosen$)
      $in_{conf} \leftarrow$ CONFLICT($created, ac_{proposed}, t_{la}$)
      if in_{conf} is False **then**
        $accepted \leftarrow$ True
        $created \cup ac_{proposed}$
      end if
    end while
  end while
  return $created$
end procedure

Algorithm 2 Data Generation algorithm for "Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning"

7. Computational experiments

In this section we will describe in detail the models used for the three most important papers produced as part of the PhD: *Spatiotemporal Graph Indicators for Air Traffic Complexity Analysis* [7]*, Towards Conflict Resolution with Deep Multiagent Reinforcement Learning* [8] and *Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning* [9]. While these works are not the only ones produced, they represent the biggest of the thesis.

Spatiotemporal Graph Indicators for Air Traffic Complexity Analysis

A correct definition of a graph requires having a set of vertices and a set of edges. In our case, vertices are the set of aircraft present in a sector at a certain time step. Therefore, we extend graph attribute to the time domain, by defining each of them per time step. The set of edges will be the interdependencies between each pair of aircraft for the time step. We define these interdependencies based on the distance between two aircraft. More concisely, if two aircraft are closer than a certain threshold, then there will be an edge between these two aircraft. The closer these aircraft are, the bigger the effect they have on each other will be, i.e. the stronger the weight of the edge connecting the pair of aircraft. If the two aircraft are in conflict, which means they are closer than the standard safety distance (5 NM horizontally and 1000 feet vertically) the effect they have on each other is maximal.

In this work, the weights are set following this rationale. We calculate horizontal and vertical distance (weight) between all pairs of aircraft. An interdependency will be added only when two aircraft are close enough horizontally and vertically. Weights are normalized to be between 0 and 1 and the final weight is the average of the horizontal and vertical interdependency. Formally, this is:

$$wh_{i,j}(t) = \begin{cases} 1 \text{ if } dh_{i,j}(t) \leq H \\ 0 \text{ if } dh_{i,j}(t) \geq thresh_h \\ \frac{thresh_h - dh_{i,j}(t)}{thresh_h - min_h} \text{ otherwise} \end{cases}$$

$$wv_{i,j}(t) = \begin{cases} 1 \text{ if } dv_{i,j}(t) \leq V \\ 0 \text{ if } dv_{i,j}(t) \geq thresh_v \\ \frac{thresh_v - dv_{i,j}(t)}{thresh_v - min_v} \text{ otherwise} \end{cases}$$

$$w_{i,j}(t) = \begin{cases} \frac{wh_{i,j}(t) + wv_{i,j}(t)}{2} \text{ if } wh_{i,j}(t) > 0 \ \ \& \ \ wv_{i,j}(t) > 0 \\ 0 \text{ otherwise} \end{cases}$$

where $wh_{i,j}(t)$ and $wv_{i,j}(t)$ are the horizontal and vertical weights at time t, $dh_{i,j}(t)$ and $dv_{i,j}(t)$ are the horizontal and vertical distance of two aircraft at time t, H and V are the horizontal and vertical safety distances and $thresh_h$ and $thresh_v$ are the horizontal and vertical thresholds. Such a definition of the interdependencies implies that they are undirected, which means that also the graph they define is undirected. Furthermore, the interdependencies are defined for a time step, therefore through their evolution in time, we are able to capture directional information such as heading. For instance, if the two aircraft that have an interdependency between them are moving towards each other, the weight of the interdependency would increase.

In this work, graph complexity and in turn sector complexity, is defined as the connectivity of the graph. Furthermore, by modelling traffic as a weighted graph, we inherently take into consideration the severity of interdependencies. There are several ways the connectivity of a graph can be

measured. Research that applies graph theory to practical problems shows that connectivity indicators that combine topological information with the weight distribution of the graph are able to provide broad and detailed information. In this work, four indicators are formally defined and illustrated: **edge density**, **strength**, **clustering coefficient** and **nearest neighbour degree**.

- **Edge density (ED)** – measures how many edges the graph has, compared to the number of edges in a fully connected graph of the same size. As we are dealing with weighted graphs, the weights are considered. Formally, ED is given as follows:

$$ED(G,t) = \frac{\sum_{(i,j)\in E} w_{i,j}(t)}{A(V_t)}, A(V_t) = \frac{|V_t|(|V_t|-1)}{2}$$

  $|V_t|$ denotes the number of vertices in the graph (i.e. the number of aircraft present in the sector) at time step t and $A(V_t)$ is the number of all possible edges. From the definition, it follows that this indicator can take values from 0 to 1. ED refers to the whole graph, and not specific vertices, making it a global connectivity measure. It relies on the concept that traffic geometries tend to be complex when there are more interdependencies between aircraft.
- **Strength** – In graph theory, the definition of strength is obtained by extending the definition of vertex degree to account for the weights of the edges. This indicator gives each aircraft its own score, and a global score is measured by taking the average of all aircraft in the graph. Formally, it is given as follows:

$$s(i,t) = \sum_{j=1}^{N} w_{i,j}(t)$$

  Strength is a natural measure of the importance or centrality of a vertex in the graph. This indicator measures the strength of the vertices in terms of the total weight of their connections. In the proposed model, it quantifies how tight interdependencies of each aircraft are. The "stronger" an aircraft is, the more interdependent it is with other aircraft, the more complex it can be considered.
- **Clustering Coefficient (CC) –** The clustering coefficient (CC) measures the local cohesiveness. This indicator provides information regarding the neighborhood of each vertex. It considers the weight of the clustered structure found in triplets. For each vertex I, CC counts the number and the weight of triplets (see: Section III) formed in the neighbourhood of i. Formally, the clustering coefficient of a vertex I, is calculated as follows:

$$CC(i,t) = \frac{\sum_{j,k}\left(w_{i,j}(t) + w_{j,k}(t)\right)}{2 \cdot (s(i,t)(\deg(i,t)-1)}, \forall (i,j,k) \in \mathcal{T}(t)$$

  where $s(i,t)$ is the strength of the current vertex, $deg(i,t)$ is the degree of the vertex at time step t and $\mathcal{T}(t)$ is the set of triplets present at time t. CC scores range from 0 to 1. If aircraft that are very tight with each other form clusters, then the situation will be more complex than if the clusters were formed by aircraft that form edges with smaller weights.
- **Nearest Neighbor Degree (NND) –** Nearest Neighbor Degree (NND) calculates a local weighted average of the nearest neighbor degree of each aircraft according to the edge weights. Formally, it is defined as:

$$NND(i,t) = \frac{\sum_{j=1}^{N} w_{i,j}(t) deg(j,t)}{s(i,t)}$$

  Such a definition implies that when edges with larger degrees are pointing to neighbors with higher degrees, the situation is more complex. Similar to Strength and CC, NND is also a local

measure, and the global measure is calculated by averaging over all vertices. The NND scores range from 0 to $|V_t| - 1$. In the case of sector complexity, the more tightly connected a neighbour of an aircraft is to other aircraft, the more likely it is for a situation to arise that requires closer monitoring or potential ATCOs interventions.

Towards Conflict Resolution with Deep Multiagent Reinforcement Learning

As mentioned, Q-learning and DQN attempt to maximize the expected value of the total reward for a given and all successive steps. However, it has been noted that this method often suffers in high dimensional action and state spaces and can fail to converge [10] [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. Formally this is given as:

$$\theta_{i+1} = \theta + \alpha \nabla J(\theta_i)$$

Such methods are commonly known as actor-critic methods. The actor uses the policy to determine which action to take, while the critic evaluates how rewarding it is to be in a certain state. According to the Policy Gradient Theorem [1] there is a direct relation between the gradient of the loss function and the gradient of the policy. This allows for improvement of the policy accordingly. These methods have several advantages over DQN. First of all, Policy Gradient methods have been found to outperform DQN methods (i.e. better convergence) in environments with stable dynamics [10]. Furthermore, these methods are inherently more effective in handling high dimensional or continuous action spaces. The predictions of DQN assign a maximum expected future reward for each possible action at each time step given a state. In cases of continuous action spaces it would not be feasible to calculate a value for each action. However, in Policy Gradient methods the parameters of the policy are adjusted directly.

Deterministic Policy Gradient (DPG) methods work in a similar way, with the distinction that actions are selected using a deterministic policy, not a stochastic one. As a drawback, if there is no noise in the action selection, exploration will be poor, which usually hinders the overall performance of a RL method. Therefore, the most effective way to use DPG is with off-policy actor-critic algorithms that learn a deterministic target policy from trajectories that have been generated by a stochastic policy.

Deep Deterministic Policy Gradient (DDPG) is an actor-critic algorithm. It uses off-policy data to learn the Q-function and uses the Q-function to learn the policy. As in DQN, DDPG uses neural networks as function approximators. Consequentially, it suffers from several of the same issues of DQN (discussed in the previous section). DDPG employs many of the same techniques to overcome the issues. Furthermore, as it is a deterministic method, exploration is added to the agent by constructing an exploration policy π'. This policy adds some noise to the actor policy:

$$\pi^{'} = \pi(s|\theta^{\pi}) + \mathcal{N}$$

Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [11], which is an extension of single agent DDPG [12], 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. 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 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.

In this work, we train two agents that represent each aircraft of the conflict pair. Figure 1

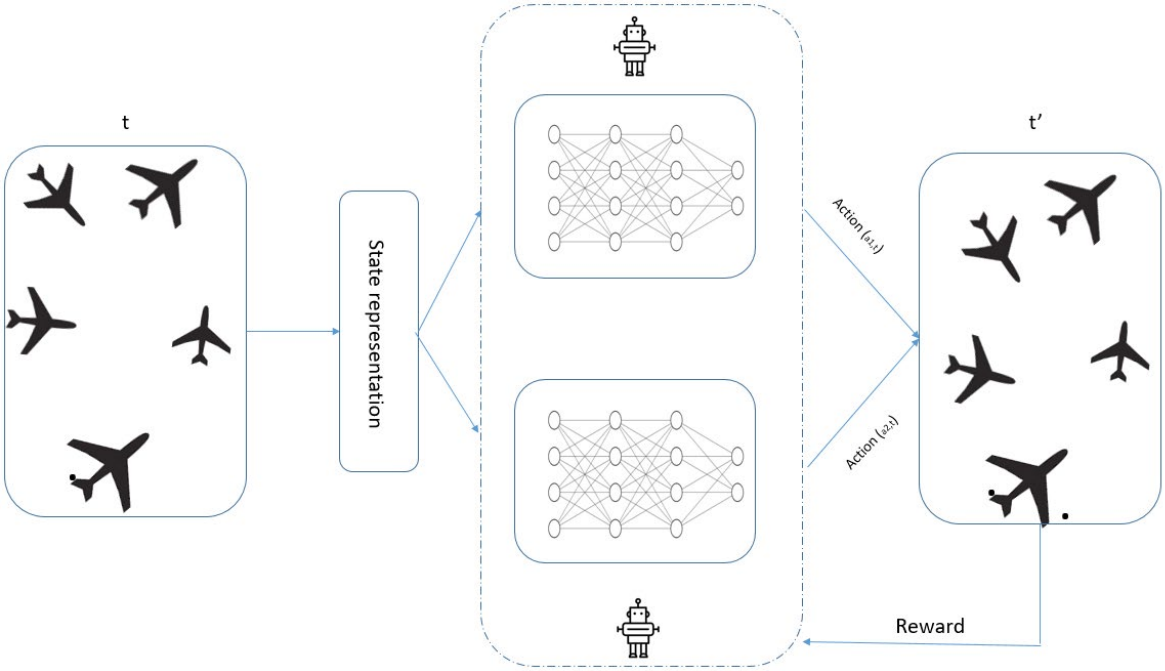

Figure 1 MADDPG for conflict resolution

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 must 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 $[v - 6\%, v + 3\%]$ [13].

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.

The reward function in this work not only is concerned with solving the conflicts, but it considers making the resolutions as efficient as possible. Thus, these factors are taken into consideration:

- Time until loss of separation and CPA – The model is encouraged to solve conflicts as soon as possible, in order to avoid dangerous situations.
- Difference from track and optimal speed – To solve conflicts through minimal manoeuvres, the agents are penalized for making big heading and speed changes.
- Fuel consumption – In order to discourage the model from taking actions that lead to big fuel consumption, we use the aircraft performance model OpenAP [14].
- New conflicts – n undesirable behaviour of CD&R 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.
- Airspace complexity – In this work we consider the complexity formulation of Koca et al. [15].

Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning

While deep learning proved effective in capturing patterns of Euclidean data, there are a number of applications where data are represented as graphs [16]. The complexity of graph data has imposed significant challenges on existing deep learning algorithms. A graph can be irregular and dynamic, as it can have a variable number of nodes and the connections between nodes can change over time. Furthermore, existing deep learning algorithms largely assume the data to be independent, which does not hold for graph data.

Recently, there was an increasing number of works that extend deep learning approaches to graph data, called Graph Neural Networks (GNNs). Variants include: Graph Attention Networks (GATs) [17],Graph Convolutional Networks (GCNs) [18] and Message Passing Neural Networks (MPNNs) [19]. We refer the reader to [16], for a comprehensive review of GNNs.

In the case of MARL, communication is often cited as a key ability for cooperative agents [20] [4]. In such a setting, agents exchange information before taking an action.

In this work, we will use Graph Convolutional Reinforcement Learning [20] (dubbed DGN by its authors), which is a GNN algorithm for cooperative agents.

In DGN, the multiagent environment is modelled as a graph G = (V, E), where V is the set of nodes and E is the set of edges. Each agent is a node and the local observation of the agent are the features of the node. Each node I has a set of B_i where $(i, j) \in E, \forall j \in B$. The set of neighbours is defined according to some criteria, depending on the environment and changes over time. In DGN, neighbor nodes can communicate with each other. Such a choice leads to the agents only considering local information when making their decisions. Another option would be to consider all agents in the environment; however, this comes with higher computational complexity.

DGN has three modules: an observation encoder, convolutional layer and Q network. The observation of an agent I at time step t, o_{i^t} is encoded into a feature vector h_{i^t} by a Multi Layer Perceptron (MLP). The convolutional layer combines the feature vectors in the local region and generates a latent feature vector h'_{i^t}. The receptive field of the agents increase by stacking more convolutional layers on top of

each other. An important property of the convolutional layer is that it should be invariant from the order of the input feature vectors. Furthermore, such a layer must be effective in learning how to abstract the relation between agents as to combine the input features.

DGN uses multihead dot-product attention [21], which is an implementation of attention which runs the attention mechanism several times in parallel, to compute interactions between agents (we refer the reader to [21]for a detailed overview of the attention mechanism). Let us denote with B_{+i} the set of neighbours B_i and agent i. The input features of the agent I are projected into query Q, key K and value V representation by every attention head. For an attention head m the relation for $i, j \in B_{+i}$ is as follows:

$$\alpha_{ij}^{m} = \frac{\exp\left(\tau \cdot W_Q^m h_i \cdot (W_K^m h_j)^T\right)}{\sum_{a \in \mathbb{B}_{+i}} \exp\left(\tau \cdot W_Q^m h_i \cdot (W_K^m h_a)^T\right))}$$

where τ is a scaling factor and W_Q^m and W_K^m are the weight matrices of the query and key for attention head m. The representations of the input features are weighted by the relation and summed together, which is done for each head m The outputs of all attention heads for an agent i are concatenated and then fed into a MLP σ as follows:

$$h_i' = \sigma\left(concatenate\left(\sum_{a \in B_{+i}} \alpha_{ij}^m W_V^m h_a \,, \forall m \in M\right)\right)$$

The graph representing the agents and the interactions between them is formalized through and adjacency matrix C, where the ith row contains a 1 for each agent in B_i and 0 for any agents not in the neighbourhood of i. The feature vectors are merged into a feature matrix F with size N x L where N is the number of agents and L is the length of the feature vector. The feature vectors in the local region of agent I are obtained by C_i x F.

The Q network in DGN is a common network as previously described. However, in DGN, the outputs of the graph convolution layer are concatenated and fed into the network. At each time step, the tuple (O, A, O', R, C) is stored in the replay buffer, where O and O' are the current and next observations, A is the set of actions, R is the set of rewards and C is the adjacency matrix. During training, a random minibatch of size S is sampled from the buffer and the loss is minimized as follows:

$$\mathcal{L}(\theta) = \frac{1}{S}\sum_{S}\frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q\left(O_{i,C,a_i}; \theta\right)\right)^2$$

Where y_i indicates the return. Another factor that can impact the training of the Q network is the dynamic nature of the graph, which can change from one time step to the other. To mitigate this, the adjacency matrix C is kept unchanged in two successive time steps when computing the Q values in training. Finally, the target network with parameters θ' is updated from the Q network with parameters θ as follows:

$$\theta' = \beta\theta + (1 - \beta)\theta'$$

where β indicates the importance of the new parameters in the target network.

In this work, we consider multi-UAV conflicts. However, multiple pairwise conflicts can have varying spatial and temporal boundaries, i.e., their overlap in space and time. Koca et al. [22], introduce the

concept of a *compound ecosystem*, with an ecosystem being the set of aircraft affected by the occurrence of a conflict. They propose that multiple ecosystems can be considered together if they have at least one common member and the conflicts overlap in time more than 10% of their duration.

For this work, we relax the requirements by not considering surrounding traffic, therefore proposing the concept of a *compound conflict*. As such, multiple pairwise conflicts can be considered collectively if and only if they share a common aircraft. We keep the temporal requirement the same as in [22].

In DGN, the nodes are the agents present in the environment. We keep the same approach by considering the UAVs as nodes in a given traffic scenario. An edge is created between two UAVs if and only if a conflict between them was detected. This choice is motivated by the fact that in DGN, agents communicate with their neighbours first and foremost. Therefore, we make this choice to facilitate cooperation between UAVs that are in conflict.

The representation of the states of the environment is one of the most critical factors that can impact the learning capability and performance of the agents. Typically, the state is formalized through a vector of a certain dimensionality, which should provide enough information to facilitate learning. Nevertheless, representations with higher dimensionality will suffer from a higher computational effort to train an effective model.

Therefore, in this work we take the state representation proposed by Isufaj et al. [8],where the state is formalized through the agents' position and speed information. More specifically the state s_i of an agent is the vector s_i = [lat,lon,hdg,spd] i.e., latitude, longitude, heading, and speed. These values are normalized into the range $[0,1]$ to make it easier for the model to be trained.

For this work, we only consider solutions through heading changes. As such, agents can choose to take one of three actions at each decision time step: turn left, turn right, do nothing, where each track change corresponds to a heading change of 15^o in either direction.

In our case, the reward consists of three terms. First, the *number of conflicts* term punishes agents according to the number of conflicts. The more conflicts the agent is in, the more it will have the incentive to solve the conflicts. Furthermore, the *deviation term* penalizes the agents for solutions that drift the agent from its original track. In this work, if an agent has deviated more than 90^o from the original route, it is penalized heavily. In cases where it has not, it is penalized as a fraction of the current deviation to the maximal deviation. This fraction is proportional to the maximal deviation. Such a term indirectly also incentivizes the agents to solve the conflicts as soon as possible, as the quicker the conflicts are solved, the less of a negative reward the agent will get. Lastly, through the *severity term*, the agents are encouraged to solve the most severe conflicts first. This term considers more severe conflicts, i.e., smaller distance at CPA, as more important to solve first. Formally, the reward function is as follows:

$$r_{i,t} = w_1 \sum_{\mathcal{E}(i)} \quad -1 + w_2 \begin{cases} -\frac{|\mu - \mu'|}{90} & \text{if } |\mu - \mu'| < 90 \\ -10 & \text{otherwise} \end{cases} + \left(-w_3\left(1 - \exp\left(1 - \frac{1}{\left(\frac{d_{cpa}}{d_{thresh}}\right)^{1/2}}\right)\right)\right)$$

Where w_1, w_2, w_3 are positive weights that indicate the importance of each term. The total reward for a given time step is the sum over all agents.

8. Results and Analysis

This section will be organized in a similar format as Section 7. The most important results of the three previous papers will be presented as they constitute the most prominent contributions of this thesis.

Spatiotemporal Graph Indicators for Air Traffic Complexity Analysis

We visualize the complexity indicators for an hour of operation. The sector we chose had on average, the most aircraft present at the same time. In Figure 2, we also visualize the occupancy in the sector at each time step. This can be thought of as an extension of this complexity metric (i.e., occupancy) in the time domain. Occupancy is one of the simplest yet most used classical complexity metrics. Here we will compare this metric with the complexity indicators outlined in Section 7.

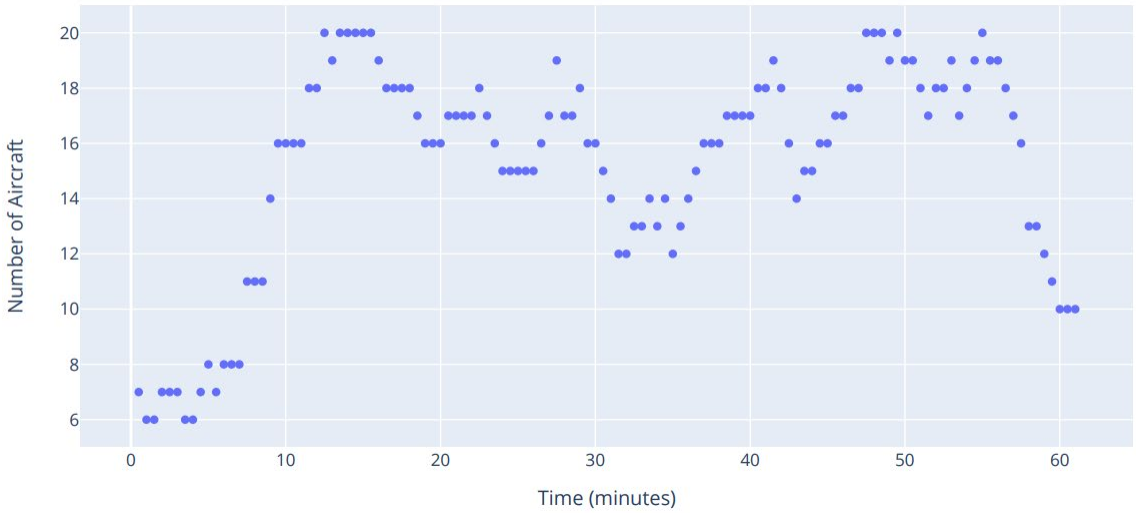

Figure 2 Number of aircraft present in the sector

In Figure 3, ED of the hour of operations is shown. The ED score is relatively low, which the highest being around 0.6, which occurs when the number of aircraft is around 8. This means that the majority of the aircraft do not get close enough to each other to be considered interdependent. This indicates that the number of aircraft does not present a full picture of complexity.

The CC score is shown in Figure 4. The overall distribution is similar to that of ED, which suggest that there is an area in the sector where aircraft tend to the cluster. Nevertheless, one can see that the clusters do not last long (around 5 minutes in this case). The NND score, shown in Figure 5 provides similar information. There is a peak around the 20 and 50 minute marks, which corresponds to a peak in the other indicators, as well as the number of aircraft. However, in the beginning, when the CC and ED scores are quite high, NND is at its lowest. This is due to a small graph size, which is further supported by the small number of aircraft. Such a behaviour is evidence of the shape of the graph during that time, meaning that even though clusters are not formed, interdependencies are not just pairwise, but span multiple aircraft. This shows that relatively complex situations can arise even when the graph is small (as evidenced by ED), with a group of aircraft needing special attention.

The strength indicator is visualized in Figure 6. Similar to the rest of the indicators, there are peaks around the 20 and 50 minute marks. However, the shape of the distribution is akin to that of NND, which supports our claim about the nature of interdependencies. On top of that, the similarity

between strength and NND further suggest that in this particular case, not only do interdependencies span multiple aircraft, they are also quite strong, which adds to the complexity of the situation.

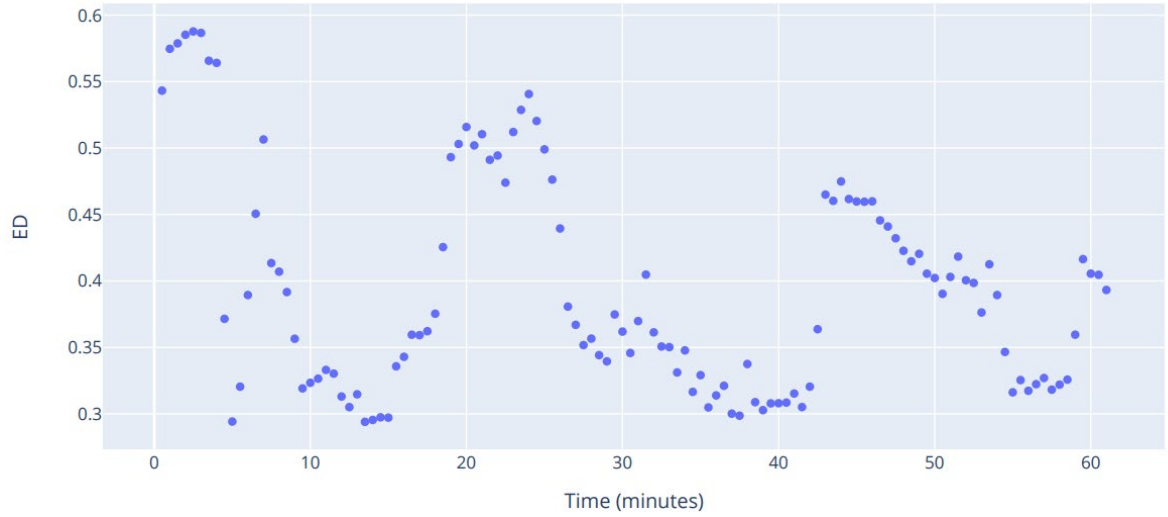

Figure 3 Edge density

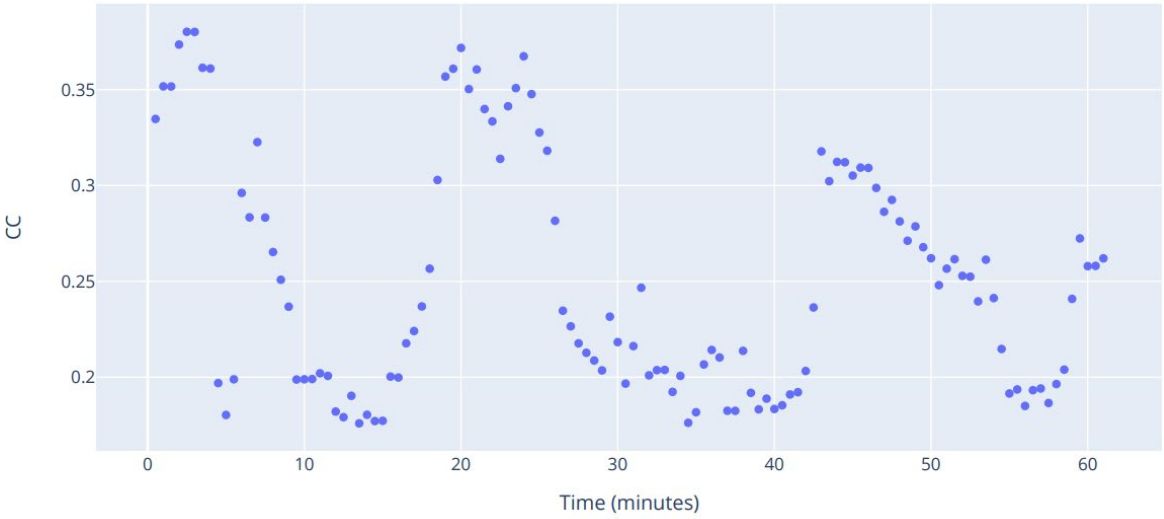

Figure 4 The clustering coefficient

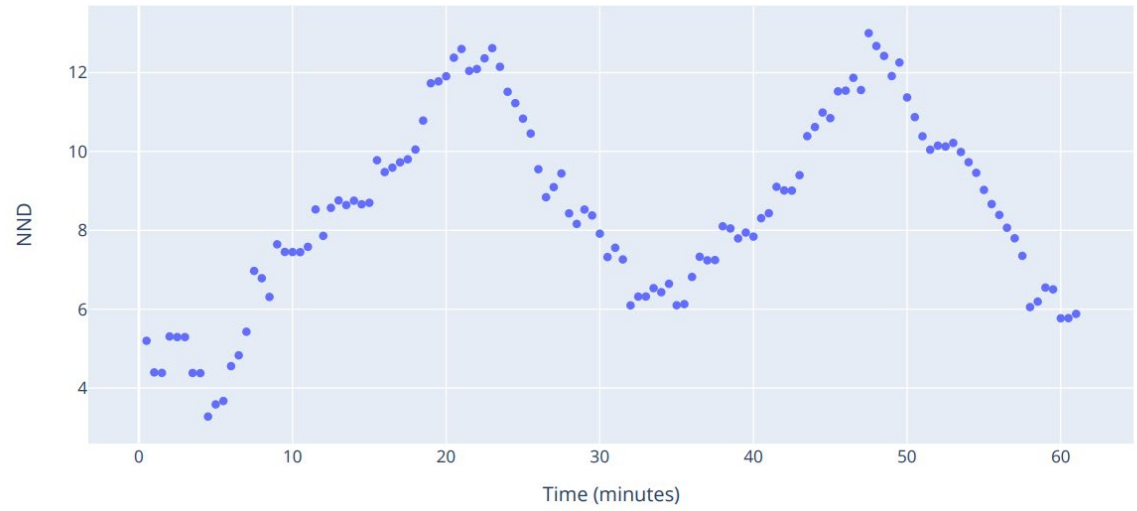

Figure 5 Nearest neighbor degree

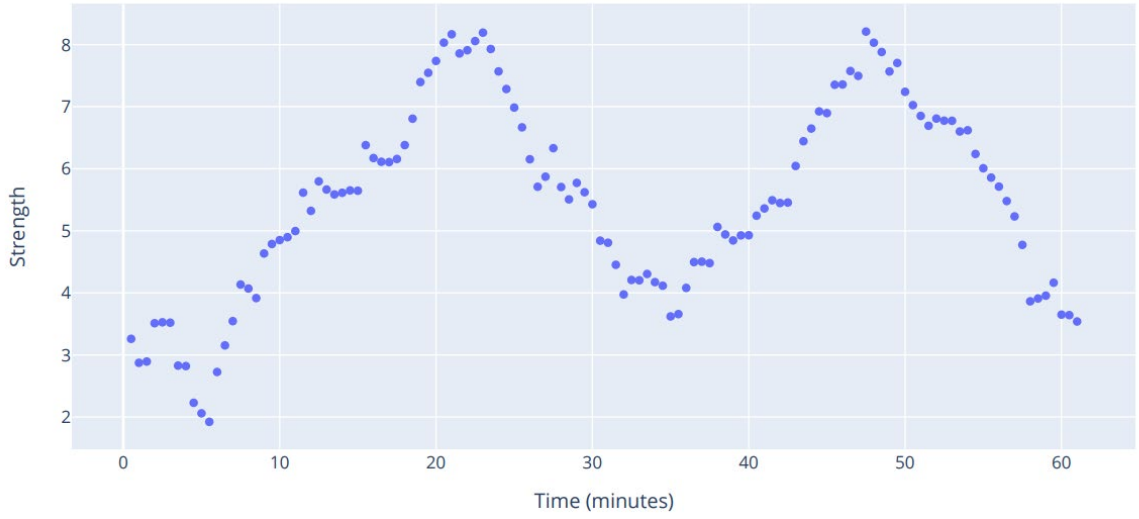

Figure 6 Strength

Towards Conflict Resolution with Deep Multiagent Reinforcement Learning

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

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

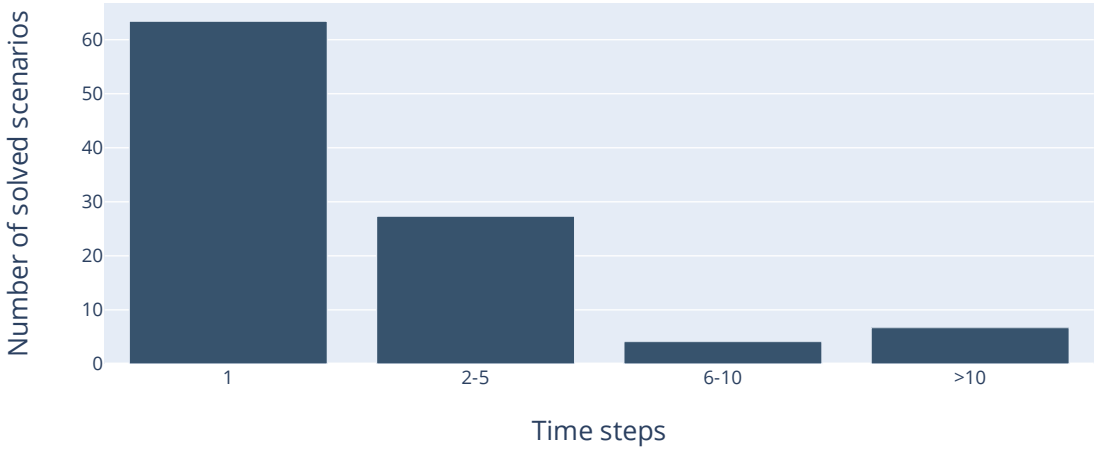

Figure 7 Number of steps required to solve the conflict. A time step is 15 seconds long.

Figure 7 shows the number of steps required to solve the conflicts. Scenarios where there were between 2 and 5, 6 and 10 steps were grouped together, while scenarios that needed 1 step and more than 10 steps were shown separately. As can be seen, 63% of solved conflicts needed only 1 time step. This is a promising result, which shows the model can learn by itself that the preferred behavior is to solve conflicts in one attempt. 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 behaviours are not preferred, its 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 around 1 minute from when the conflict was detected.

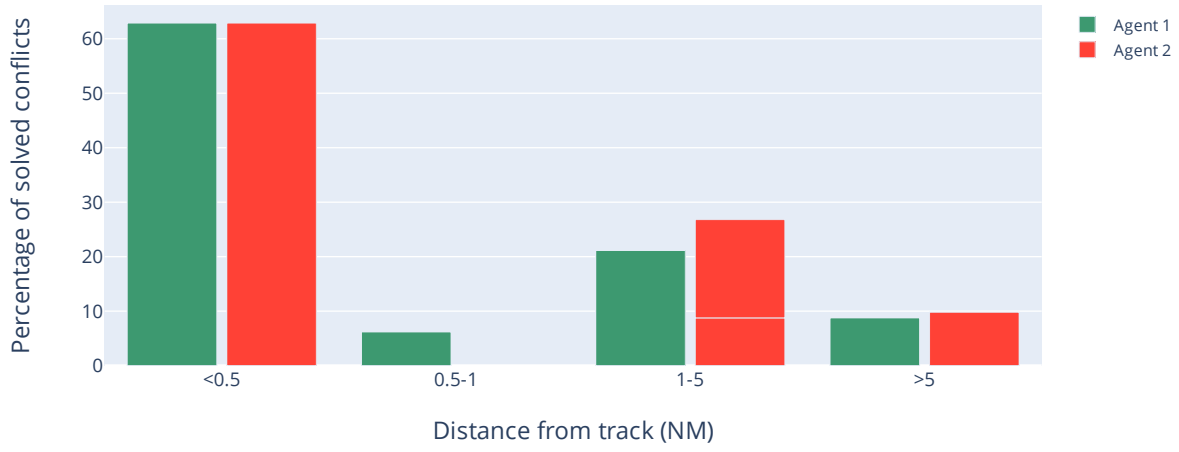

Figure 8 Distance of each agent from track in nautical miles when the conflict is resolved

Figure 8 shows the distance from track at the end of the conflict for each agent. As one can seem both agents can solve around 65% of conflicts within 0.5NM of their original track and around 70% within 1NM. This result is promising, as resolutions that minimally displace the aircraft from their track are preferred. Both agents show similar performance, which indicates the resolution would be likely

acceptable by both aircraft. However, in this work we assume cooperative agents, which cannot always be accepted in practice. We calculated the Pearson correlation coefficient between number of time steps required to solve the conflict and finals distance from track. The result was 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 it, this resolution cannot be accepted in practice, as it would require a huge deviation, which will result in major delays.

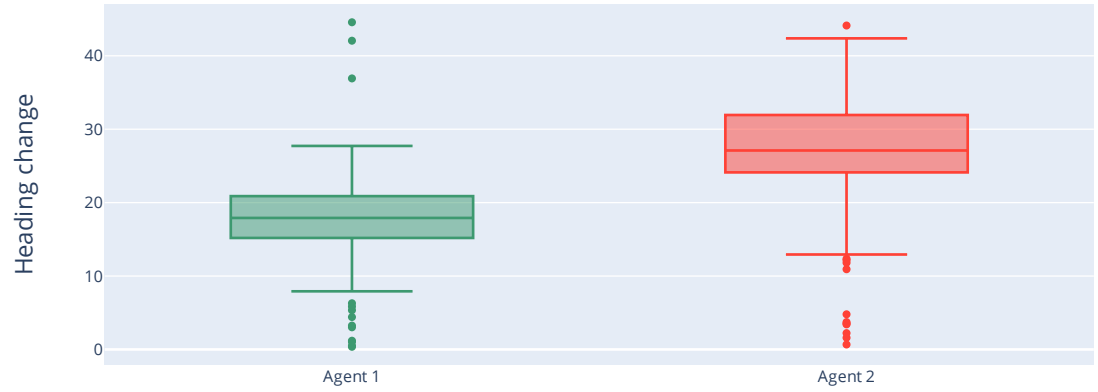

Figure 9 Average heading change needed to solve the conflicts

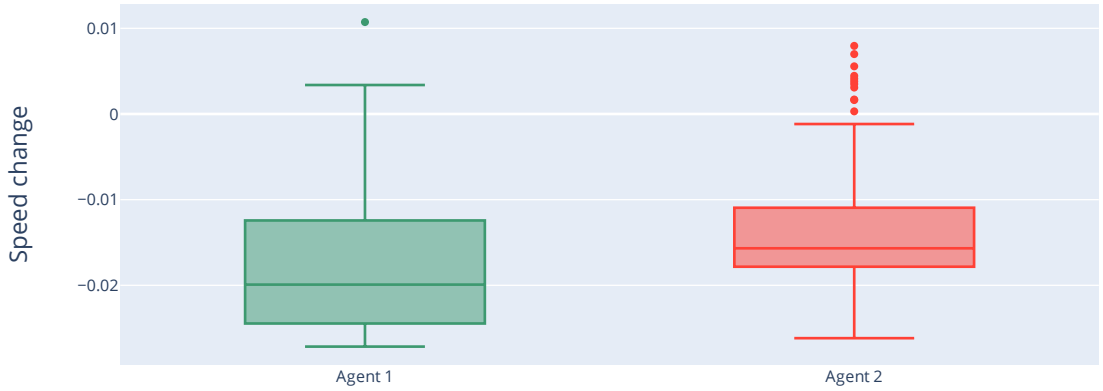

Figure 10 Average speed change needed to solve the conflicts

As mentioned previously, at each time step the agents could make a heading change of at most $\pm 45^{o}$ and a speed change in the range $[v - 6\%, v + 3\%]$. Figure 9 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 e solved, the average of all time steps is shown. Figure 10 shows the same information with respect to the speed changes, with changes being the difference in percentage to 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 have an average decrease of -0.01%. This is a relevant result, as speed changes in this time frame are known to be not as efficient 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 the changes decrease speed, which can be an indication from the performance model that the optimal speed can be improved, however this would need further investigation.

For heading changes, the results show that 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 take only once by each agent. This further shows the effect of the reward function in the behavior of the agents, as each of them prefers small changes that solve the conflict quickly. However, we note that Agent 2 makes on average a bigger heading change. The fact that the majority of existing conflicts are solved only in one time step shows that the learned behaviour 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 desirable behaviors as ATCOs do not issue manoeuvres that create new conflicts.

Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning

In this work we train the model with compound conflicts with 3 and 4 agents. First the model was trained for 10,000 episodes with scenarios of compound conflicts with 3 UAVs. Then it was trained for a further 10,000 episodes with scenarios of compound conflicts with 4 UAVs. In this way, we utilize the learned policies of the previous agents to fine-tune them in the four-agent case and train the new agent from scratch. The models were trained for around 10 hours each.

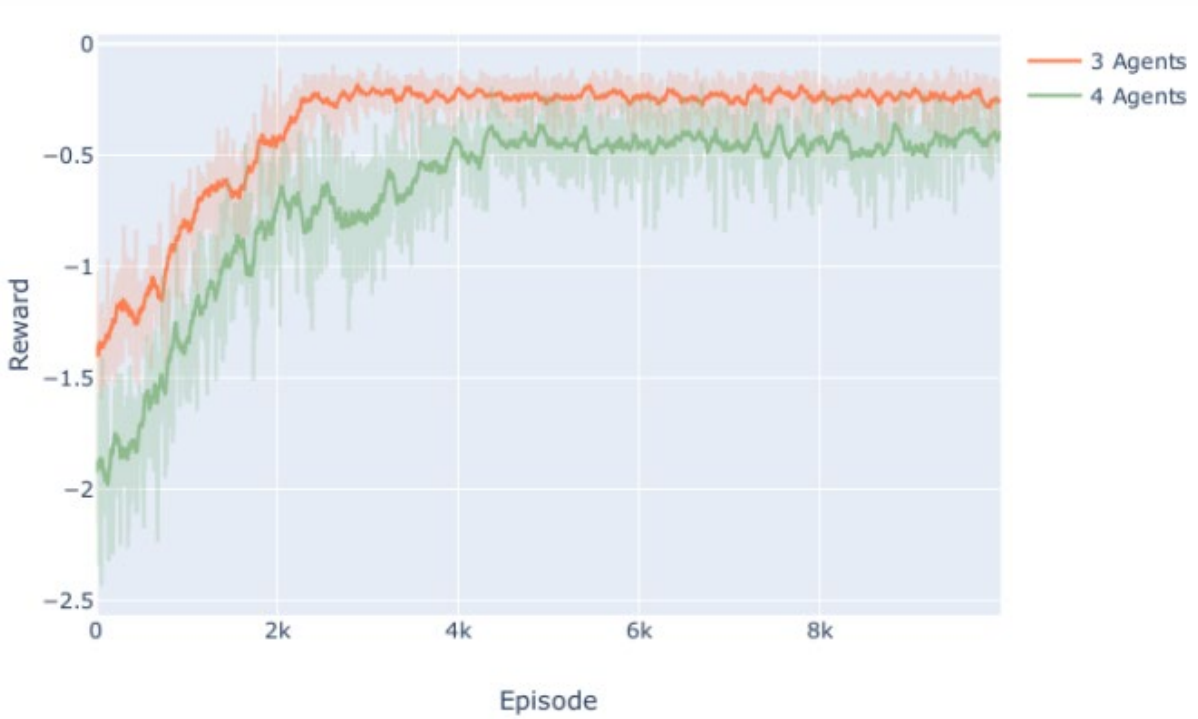

Figure 11 Evolution of cumulative reward per episode

Figure 11 shows the evolution of cumulative reward for both cases. As the agents are cooperative, we are interested in the overall reward gained per episode and do not concern ourselves with the individual rewards. In this work, we utilize negative rewards, so the maximum that the agents can get

is 0. In the case if 4 agents, the reward seems a bit lower, however this comes as a result of there being more agents present, which takes actions to solve the conflicts thus inflicting itself some negative rewards for going away from track. According to the figure, the model converges on both occasions. This means that the agents are successfully able to improve their policies with gained experience. However, in the case 3 present agents, convergence happens around 2000 episodes, while 4000 episodes are required for the 4 agent case. In this latter scenario, there are more possible scenarios that can be generated, therefore increasing the variance of situations that the agents are presented with. Furthermore, in the beginning of training, the already present agents employ their learned policies, while the new agent is exploring the possible actions, which reduces the overall reward the agents get.

Figure 12 Number of losses of separation in comparison with the average unmitigated case

In Figure 12, the number of LOSS is shown. The number of LOSS of the average unmitigated case (for both 3 and 4 agents) is shown with the dashed line. The reward performance translates directly to successfully avoiding LOOs. In the case with 3 agents, after convergence the average LOSS per episode Is less than 1. This indicates that the agents are able to successfully solve conflicts before violating the self-separation distance. In the case of 4 agents, the average is 1 LOSS per episode. However, through our results we note that the models manage to always avoid near misses in both cases, as the NMAC distance is never breached.

The results shown so far indicate that the agents manage to successfully learn how to solve the task. The model converges fast and maintains its knowledge of the system, thus avoiding the common forgetting issues. However, it is important to understand what strategies they learned. This information is shown in Figures 13 and 14. For the sake of simplicity, we only show the frequency of actions for the last 200 episodes. We note that in both settings, the agents take the *go left* action in the majority of cases. While the direction of the action might not be as important, the learned strategy suggests that agents take the same action. This results in agents increasing the distance between them, as taking the same action head-on or crossing scenarios results in them going in different directions. However, in overtaking scenarios such a strategy does not immediately solve the conflict. Nevertheless, through the reward agents must learn that the conflict with the smallest CPA distance

is the most urgent. As such, it can happen that agents prefer to delay the solution in an overtaking scenario, by taking several small changes in the same direction. While this is not immediately desirable, attempting to make a heading change to the opposite direction could create a more severe conflict with the head-on or crossing agents.

Figure 13 Frequency of actions for last 200 episodes of compound conflict with 3 agents

Figure 14 Frequency of actions for last 200 episodes of compound conflict with 4 agents.

In this work, we do not put any restrictions to the agents and do not inject expert knowledge in them, thus they start learning from a blank state. The results show that the agents are able to learn a strategy that successfully solves the compound conflicts in scenarios with 3 and 4 agents.

9. Conclusions and look ahead

The goal of this PhD project was divided into two clear directions: investigate how machine learning can be applied to safety in aviation and define complexity in a way that challenges current methods and overcomes their shortcomings.

Our first major contribution has been in modelling air traffic (be that manned or unmanned) as a graph. We formalized four indicators based on graph theory to measure the complexity of an airspace. These indicators combine topological information gathered from interdependency geometries with the severity of interdependencies to present a full and nuanced picture of complexity. Furthermore, we consider the evolution of complexity in time and do not simply give one single sector complexity score. Simulation results indicated that the indicators we propose give detailed information and overcome drawbacks of existing metrics. We evaluated our approach using real and synthetic traffic and demonstrated that the indicators express different facets of complexity, confirming that all indicators are needed. The way we define complexity also provides a new framework in the design of conflict resolution algorithms which considers the reduction of airspace complexity in addition to safety preservations. CR algorithms could be discouraged from providing solutions that increase the overall complexity of the airspace.

By modelling traffic as a graph, we have opened the door to further applications of graph theory in aviation. Our work should be the baseline for designing more complexity indicators based on graph theory. As such, future work should consider different methods for the definitions of interdependencies and the value of thresholds. Finally, a very important continuation should be the mapping between the indicators and controller workload, which is an ongoing topic in the ATM community. Such work should investigate if the proposed indicators are a good predictor of the measured workload. Last but not least, it is very important to consider how the information provided by the indicators should be presented to the controllers. In our work, we make a simple attempt by showing the interdependencies and the indicator scores at various points in time, however more research is needed which takes into account controller preferences and other factors.

In this thesis, we model conflict resolution as a MARL problem. In such a setting, aircraft are agents that are capable of making decisions and learning from their experiences. To the best of our knowledge, our work is one of the first attempts to uses a MARL approach to conflict resolution. We started our work by tackling pairwise conflicts at the tactical level for en-route traffic.

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. Each scenario lasts 20 minutes, in which each aircraft within the conflict pair is assigned an agent taking actions every 15 seconds. In this work, agents are able to handle continuous actions space, which means that we do not prescribe fixed manoeuvres to solve the conflict. This overcomes a common limitation of existing research, where fixed manoeuvres are usually issued.

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 heading changes smaller than the

allowed maximum. 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.

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. 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 manoeuvres. 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 conflict 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.

To overcome the major limitations of the previous approach, namely scalability and the nonstationarity of the environment, we utilize GNNs. They are a recent advance in deep learning which have proved very effective in non-euclidean data. To utilize GNNs in a MARL setting, we use Graph Convolutional Reinforcement Learning. We tackle multi-UAV conflict resolution with cooperative agents in situations with 3 and 4 present agents. Air traffic is represented as a graph with aircraft as nodes. An edge is created between every two aircraft in a pairwise conflict. We use graph convolutional reinforcement learning, which provides a communication mechanism between connected agents. This means that conflicting aircraft are allowed to communicate with each other and develop cooperative strategies. To formally define a multi-UAV conflict, we propose the concept of compound conflicts, which are conflicts that have tight spatial and temporal boundaries. Results show that the agents are able to improve their policies and thus solve the task. For both settings, we observe an improvement both in number of LOSS present and duration of LOSS with the majority of scenarios after convergence having no LOSS (i.e., the compound conflict is solved). Furthermore, the agents are able to discover a strategy that increases the overall distance between them. As such, they effectively learn to solve the most severe conflicts first and then solve the remaining conflicts while making sure that no new conflicts are created.

However, there are several aspects that must be further researched. For instance, in this work we use a maximum of 4 agents in the scenario. In reality, the number of agents in a compound conflict can not be always decided beforehand, thus a solution that adapts to N agents must be sought. Furthermore, the reward function could be further elaborated to include terms that deal with the quality of solutions, such as optimizing for battery usage or number of actions taken. Finally, the action space can be extended to include solutions by speed or altitude changes.

Furthermore, we are in the process of investigating some questions that have arisen from our research in this PhD project. First of all, we are investigating practical ways the airspace complexity indicators can be used. As a first step, we are in the process of identifying high complexity spatio-temporal areas in a sector. To do so, we make use of the choice to model air traffic as a graph. We are evaluating different community detection algorithms in order to show if are able to identify communities in air traffic. The assumption that we make is that these communities correspond to high complexity areas. After this assumption is verified, similar methods as in [9] will be used to improve the complexity of these areas using MARL with GNNs. In another project, the scalability of the CR method presented in [9] is being tested. Moreover, the correlation between conflicts and the complexity indicators proposed in [7] is being studied. To do so, large scale and high density scenarios are being generated. The candidate was the driving force behind improving the efficiency of these simulations through the use of parallel computing. Finally, we are investigating more fundamental questions in MARL, such as agent coordination, heterogeneous agents and transparency. If we assume a large enough airspace, agents can have individual goals, in addition to some common goal. For instance, imagine a fleet of drones that must deliver goods to their destination (i.e. individual goal), but also have to maintain certain safety distances from other agents or static obstacles (i.e. common goal). In such situations, agents do not have to cooperate at all times. There could be only a few situations where they would have to coordinate. Therefore, it is important to investigate how agents can learn to coordinate and what mechanisms they can use to facilitate coordination. So far, agents have been assumed to be homogeneous, which means that they all have the same physical attributes, can take the same actions and all are benevolent. However, this assumption might not hold in cases with a large number of agents (similar to the first research question). Therefore, we will investigate how the presence of heterogeneous agents (e.g. malicious intent) can affect learning and overall performance.

Lastly, as the application field is safety critical, it is important to head towards achieving the acceptability of the models from the human controllers (i.e. air traffic controllers). Therefore, ways of introducing transparency and interpretability into the developed models will be investigated. These questions are being investigated in the context of dynamic re-routing for UAS. This setting provides a natural extension to current work of the candidate, as longer time-frames and a larger number of agents can be considered in a realistic application.

Industrialisation

We believe the work performed in this PhD project has the potential to be used in industry in two different contexts: airspace complexity and MARL applications. For the former, our previous results and current work has shown that graph applications in airspace complexity can give a high degree of granular information (up to the level of single aircraft) while being fast. Furthermore, they can give real-time results even in a modest machine. On the other hand, through our results, we can observe that MARL can tackle problems in aviation from different perspectives. While in this PhD project, we have envisioned our work to be in the ground, there is no inherent limitation to where it could be actually deployed, however adaptions would have to be made. In the case of MARL, there is a considerably higher computational burden to be considered. Training a model can take several days,

but once trained, deploying it and getting a response from the model can be done in real time. Training is usually conducted in dedicated servers, which are also offered by companies such as Google or Amazon. Indeed, crucial issues such as transparency, safety guarantees and quality of the model are issues that need further investigating.

For both directions, significant effort has to be made to validate and verify the approaches extensively in order to make them compatible with existing requirements and infrastructure. Furthermore, there are important questions to be answered in terms of user friendliness and what would be the best way to present the information to the users.

10. References

10.1 Link to PhD thesis / repository

This thesis will be published at UAB's online repository which can be found following this link: https://www.uab.cat/libraries/

10.2 Associated outputs and publications

Koca T., Isufaj R., Piera MA., "Strategies to Mitigate Tight Spatial Bounds Between Conflicts in Dense Traffic Situations", 9th SESAR Innovation Days, Athens, Greece, 2019. https://www.sesarju.eu/sites/default/files/documents/sid/2019/papers/SIDs_2019_paper_82.pdf

Isufaj R, Koca T, Piera MA. "Spatiotemporal Graph Indicators for Air Traffic Complexity Analysis". *Aerospace*. 2021; 8(12):364. https://www.mdpi.com/2226-4310/8/12/364

Isufaj R, Aranega Sebastia D, Piera MA. "Towards Conflict Resolution with Deep Multi-Agent Reinforcement Learning", in 14th USA/Europe Air Traffic Management Research and Development Seminar (ATM2021), 2021 (*Best Paper Award in the "Separation" track. To appear also in Journal of Air Transportation)*

Isufaj, R., Omeri, M., & Piera, M. A. (2022). Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning. *Applied Sciences*, *12*(2), 610. https://www.mdpi.com/2076-3417/12/2/610/htm

Kola I., Isufaj R., & Jonker C. Does Personalization Help? Predicting How Social Situations Affect Personal Values. *To Appear in Proceedings of the First International Conference on Hybrid Human-Machine Intelligence (July 2022)*

Omeri, M., Isufaj, R. and ORTIZ, R.M., Quantifying Well Clear for autonomous small UAS.(*Currently under review in IEEE Access)* *https://www.researchgate.net/profile/Marsel-*

Omeri/publication/358982827_Quantifying_Well_Clear_for_autonomous_small_UAS/links/6220d3f219d1945aced2e27d/Quantifying-Well-Clear-for-autonomous-small-UAS.pdf

Presentations in each of the three Engage KTN summer schools

https://engagektn.com/summer-school-2019/

https://engagektn.com/summer-school-2020/

https://engagektn.com/summer-school-2021/

10.3 References cited in this report

[1] R. S. Sutton, A. G. Barto and others, Introduction to reinforcement learning, vol. 135, MIT press Cambridge, 1998.

[2] R. Bellman, "Dynamic programming and stochastic control processes," *Information and control,* vol. 1, p. 228–239, 1958.

[3] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare and J. Pineau, "An introduction to deep reinforcement learning," *arXiv preprint arXiv:1811.12560,* 2018.

[4] R. Dalmau and E. Allard, "Air Traffic Control Using Message Passing Neural Networks and Multi-Agent Reinforcement Learning".

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602,* 2013.

[6] J. M. Hoekstra and J. Ellerbroek, "Bluesky ATC simulator project: an open data and open source approach," in *Proceedings of the 7th International Conference on Research in Air Transportation*, 2016.

[7] R. Isufaj, T. Koca and M. A. Piera, "Spatiotemporal graph indicators for air traffic complexity analysis," *Aerospace,* vol. 8, p. 364, 2021.

[8] R. Isufaj, D. Aranega Sebastia and M. A. Piera, "Towards Conflict Resolution with Deep Multi-Agent Reinforcement Learning," in *14th USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)*, 2021.

[9] R. Isufaj, M. Omeri and M. A. Piera, "Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning," *Applied Sciences,* vol. 12, p. 610, 2022.

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275,* 2017.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971,* 2015.

[13] R. Breil, D. Delahaye, L. Lapasset and É. Féron, "Multi-agent Systems for Air Traffic Conflicts Resolution by Local Speed Regulation," 2016.

[14] J. Sun, J. M. Hoekstra and J. Ellerbroek, "OpenAP: An open-source aircraft performance model for air transportation studies and simulations," *Aerospace,* vol. 7, p. 104, 2020.

[15] T. Koca, M. A. Piera and M. Radanovic, "A Methodology to Perform Air Traffic Complexity Analysis Based on Spatio-Temporal Regions Constructed Around Aircraft Conflicts," *IEEE Access,* vol. 7, p. 104528–104541, 2019.

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems,* vol. 32, p. 4–24, 2020.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903,* 2017.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907,* 2016.

[19] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, 2017.

[20] J. Jiang, C. Dun, T. Huang and Z. Lu, "Graph convolutional reinforcement learning," *arXiv preprint arXiv:1810.09202,* 2018.

[21] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart and others, "Relational deep reinforcement learning," *arXiv preprint arXiv:1806.01830,* 2018.

[22] T. Koca, R. Isufaj and M. A. Piera, "Strategies to Mitigate Tight Spatial Bounds Between Conflicts in Dense Traffic Situations".

[23] Zaoli, S., Scaini, G., & Castelli, L. (2021). Community Detection for Air Traffic Networks and Its Application in Strategic Flight Planning. Sustainability, 13(16), 8924.

Annex I: Acronyms

Term	**Definition**
ATM	Air Traffic Management
CC	Clustering Coefficient
CD	Conflict Detection
CD & R	Conflict Detection and Resolution
CPA	Closest Point of Approach
CR	Conflict Resolution
DGN	Graph Convolutional Reinforcement Learning
DPG	Deterministic Policy Gradient
DQN	Deep Q Networks
ED	Edge Density
GAT	Graph Attention Network

Term	Definition
GCN	Graph Convolution Network
GNN	Graph Neural Network
LOSS	Loss of separation
MADDPG	Multiagent Deep Deterministic Policy Gradient
MARL	Multiagent Reinforcement Learning
MDP	Markov Decision Process
ML	Machine Learning
MPNN	Message Passing Neural Network
NMAC	Near Mid-Air Collision
NND	Nearest Neighbour Degree
RL	Reinforcement Learning
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle

-END OF DOCUMENT-