Universitat Autònoma de Barcelona

Towards Unified Air Traffic Complexity Management through Graph Theory and Artificial Intelligence

Ralvi Isufaj

Towards Unified Air Traffic Complexity Management through Graph Theory and Artificial Intelligence

Doctoral Thesis by

Ralvi Isufaj

Presented in Partial Fulfillment of the Requirements for the
PhD Degree in Electronic and Telecommunication Engineering

Supervised by: Dr. Miquel Angel Piera

AUTONOMOUS UNIVERSITY OF BARCELONA
Sabadell, Barcelona, Spain

2022

To Salvör

ACKNOWLEDGEMENTS

I would like to thank first and foremost my supervisor Dr. Miquel Angel Piera, whose knowledge, dedication and support have been crucial in completing this project. I will always be thankful for providing me this opportunity, allowing me to express my ideas and make mistakes and will cherish and miss our long and fervent discussions. His roles exceeded that of just a supervisor and I feel fortunate to consider him my mentor.
This PhD project would have not been possible without the generous funding of SESAR Joint Undertaking under the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 783287.
I am very grateful to my current and former colleagues at UAB. Our collaborations, discussions as well as support and friendship have helped push through in difficult times. In particular, I would like to thank Dr. Thimjo Koca for the articles we worked together as well as for helping me settle in Barcelona and being there for me. Special thanks go to Marsel Omeri for our collaboration in the field of UAS, giving me a hand at improving in table tennis and our (let's be honest here, very frequent) coffee breaks. They made my time at UAB much more special.
Many thanks to Dr. Luciano Cavalcante Siebert for providing me the opportunity to have a research stay at TU Delft where I saw the importance of the human in Artificial Intelligence. I would also like to thank Dr. Christian Eduardo Verdonk Gallego from CRIDA, whose guidance in the latter stages of this thesis improved the quality of my work.
I would like to extend my sincere thanks to David, Javier and Jaume from UAB and Lichen from TU Delft who entrusted me to be (co-supervisor) of their respective theses. You guys pushed me to improve on a professional and personal level.
I am also thankful to Ilir Kola, who as a friend and colleague has always been there to discuss, share ideas and support as well as to just talk when I needed to. I express my thanks to Flori, mainly for disagreeing about who the G.O.A.T. is.
Lastly, I would like to thank my family and loved ones. To mom and dad, without whose love, support and sacrifice I would not be here. To Shirla, who made me an uncle and to baby Nalta, through whom I have learned unconditional love. And finally, to Salvör whose endless love, support and patience have shaped me to who I am today. I hope I made you all proud.

“And once the storm is over, you won’t remember how you made it through, how you managed to survive.
You won’t even be sure, whether the storm is really over.
But one thing is certain. When you come out of the storm, you won’t be the same person who walked in.”

HARUKI MURAKAMI

ABSTRACT

Air Traffic Management (ATM) is formally defined as the dynamic, integrated management of air traffic and airspace safely, economically and efficiently, through the provision of facilities and seamless services in collaboration with all parties and involving airborne and ground-based functions. This implies a complex socio-technical system with several layers and sub-systems. The performance of these layers is measured through various Key Performance Areas (KPAs), of which some of the most important are safety, capacity, cost-efficiency and the environmental impact. Although part of ATM structure, each of the layers have different objectives which in practice compete to maximize their own goals. A similar situation can be expected for Unmanned Aerial Systems (UAS), where the development of different UAS traffic management systems is being guided with a similar approach to that of ATM. In this thesis, we envision air traffic complexity to be the framework through which a common understanding among stakeholders making decisions at the different ATM layers is enhanced. Furthermore, we focus on automated Conflict Detection and Resolution (CD & R) to investigate how Multi-Agent Reinforcement Learning (MARL) can facilitate the progress to autonomous ATM and UAS traffic management systems.

To achieve these goals, we first define air traffic complexity in such a way that provides elaborate information efficiently to different stakeholders. As an initial step, we provide a generic definition of pairwise interdependencies between aircraft, which is based on the distance at a time step and is adaptive to the use-case and context. We use graph theory to model air traffic as a dynamic graph and define four complexity indicators that combine different topological information and the severity of the interdependencies to give a detailed and nuanced evolution of traffic complexity in a certain airspace. These indicators are extended to introduce the concept of single aircraft complexity, through which complex spatio-temporal areas in a given airspace are identified. Moreover, we investigate the effectiveness of these complexity indicators in the domain of UAS, to show how the same definition of air traffic complexity can be used in different domains.

Focusing on CD & R, we approach this problem through MARL, which is a paradigm of Machine Learning (ML) where multiple agents interact with an environment and themselves to maximize some notion of accumulated reward. We first extend existing work by proposing a model that not only solves conflicts but also show that it is possible to consider several other factors that affect efficiency and the environmental impact. This approach is further enhanced by the use of Graph Neural Networks (GNNs), which facilitate cooperation and communication. We show that through cooperation it is possible for autonomous agents to learn resolution strategies that are similar to known strategies by human controllers, which existing work using GNNs has not been able to achieve.

While these two goals might seem separate, in this thesis we argue that a unified air traffic complexity management system with the application of AI can move aviation further to its quest for autonomy. This thesis is concluded with a vision for AI in aviation that considers meaningful human control and value alignment as the most imperative directions for future research.

RESUMEN

La Gestión del Tráfico Aéreo (ATM) se define formalmente como la gestión dinámica e integrada del tráfico y el espacio aéreo de forma segura, económica y eficiente, mediante la disposición de infraestructuras y servicios integrados en colaboración con todas las partes e involucrando funcionalidades en el lado aire y en el lado tierra. Se trata de un sistema socio-técnico complejo estructurado en varios niveles y subsistemas interrelacionados. El rendimiento en cada nivel se mide a través de varias áreas clave de rendimiento (KPA), de entre las cuales, algunas de las más importantes son la seguridad, la capacidad, la rentabilidad y el impacto ambiental. Aunque forman parte de la misma estructura ATM, cada una de los niveles tiene diferentes objetivos que en la práctica compiten entre sí, para maximizar sus propias metas. Se espera un escenario similar para los sistemas aéreos no tripulados (UAS), donde el desarrollo de diferentes sistemas de gestión de tráfico de UAS obedecen a un enfoque similar al de ATM. En esta tesis, concebimos la complejidad del tráfico aéreo como el marco de referencia para tener una comprensión única y compartida entre los actores que toman decisiones en los diferentes niveles del sistema ATM. Por otro lado, también hemos trabajado en la Detección y Resolución de Conflictos (CD & R) automatizada para investigar cómo el Aprendizaje por Refuerzo de Múltiples Agentes (MARL) puede facilitar avanzar hacia los sistemas autónomos de gestión de tráfico ATM y UAS.

Para abordar ambos objetivos, primero definimos la complejidad del tráfico aéreo de tal manera que proporcione eficientemente una información elaborada a los diferentes actores interesados. Como primer paso, proporcionamos una definición genérica de las interdependencias entre pares de aeronaves, que se basa en la distancia en tiempos discretizados y se adapta al caso de uso y al contexto. Hemos utilizado la teoría de grafos para modelar el tráfico aéreo como un gráfico dinámico y definimos cuatro indicadores de complejidad que combinan diferente información topológica y la severidad de las interdependencias para dar una evolución detallada y peculiar de la complejidad del tráfico en un espacio aéreo determinado. Estos indicadores se han ampliado para introducir el concepto de complejidad de aeronave, a través del cual se identifican áreas espacio-temporales complejas en un espacio aéreo determinado. Además, se ha investigado la efectividad de estos indicadores de complejidad en el dominio de los UAS, para mostrar cómo la misma definición de complejidad del tráfico aéreo se puede utilizar en diferentes dominios.

Centrándonos en el CD & R, abordamos este problema a través de MARL, que es un paradigma de Aprendizaje Automático (ML) donde múltiples agentes interactúan con el entorno y entre ellos mismos para maximizar alguna noción de recompensa acumulada. Primero ampliamos el trabajo existente al proponer un modelo que no solo resuelve conflictos sino que también muestra que es posible considerar diversos factores que afectan la eficiencia y el impacto ambiental. Este enfoque se ve reforzado por el uso de Graph Neural Networks (GNN), que facilitan la cooperación y la comunicación. Mostramos que a través de la cooperación es posible que los agentes autónomos aprendan estrategias de resolución que son similares a las estrategias conocidas por los controladores humanos. Dichos resultados no habían sido reportados en la literatura de GNN.

Si bien estos dos objetivos pueden parecer separados, en esta tesis argumentamos que un sistema unificado de gestión de la complejidad del tráfico aéreo con aplicaciones de la IA puede contribuir a avanzar mas en los sistemas autónomos en el sector aeronáutico. Esta tesis concluye con una visión de la IA en la aviación que considera significativo el humano en el lazo de control y la alineación de valores como esencia de las líneas de investigación futura.

RESUM

La Gestió del Trànsit Aeri (ATM) es defineix formalment com la gestió dinàmica i integrada del trànsit i l'espai aeri de forma segura, econòmica i eficient, mitjançant la disposició d'infraestructures i serveis integrats en col·laboració amb totes les parts i involucrant-hi funcionalitats del costat aire i del costat terra. Es tracta d'un sistema socio-tècnic complex estructurat en diferents nivells i subsistemes interrelacionats. El rendiment a cada nivell es mesura a través de diverses àrees clau de rendiment (KPA), d'entre les quals, algunes de les més importants són la seguretat, la capacitat, la rendibilitat i l'impacte ambiental. Tot i que formen part de la mateixa estructura ATM, cadascun dels nivells té diferents objectius que a la pràctica competeixen entre si, per maximitzar les seves pròpies metes. S'espera un escenari similar per als sistemes aeris no tripulats (UAS), on el desenvolupament de diferents sistemes de gestió de trànsit d'UAS obeeixen a un enfocament semblant al d'ATM. En aquesta tesi, concebem la complexitat del trànsit aeri com el marc de referència per tenir una comprensió única i compartida entre els actors que prenen decisions als diferents nivells del sistema ATM. D'altra banda, també hem treballat en la Detecció i Resolució de Conflictes (CD & R) automatitzada per investigar com l'Aprenentatge per Reforç de Múltiples Agents (MARL) pot facilitar avançar cap als sistemes autònoms de gestió de trànsit ATM i UAS.

Per abordar tots dos objectius, primer definim la complexitat del trànsit aeri de manera que proporcioni eficientment una informació elaborada als diferents actors interessats. Com a primer pas, proporcionem una definició genèrica de les interdependències entre parells d'aeronaus, que es basa en la distància en temps discretitzats i s'adapta al cas d'ús i context. Hem utilitzat la teoria de grafs per modelar el trànsit aeri com un gràfic dinàmic i definim quatre indicadors de complexitat que combinen diferent informació topològica i la severitat de les interdependències per donar una evolució detallada i peculiar de la complexitat del trànsit en un espai aeri determinat. Aquests indicadors han estat ampliats per introduir el concepte de complexitat d'aeronau, a través del qual s'identifiquen àrees espai-temporals complexes en un espai aeri determinat. A més, s'ha investigat l'efectivitat d'aquests indicadors de complexitat al domini dels UAS, per mostrar com la mateixa definició de complexitat del trànsit aeri es pot utilitzar en diferents dominis.

Centrant-nos en CD & R, abordem aquest problema a través de MARL, que és un paradigma d'aprenentatge automàtic (ML) on múltiples agents interactuen amb l'entorn i entre ells mateixos per maximitzar alguna noció de recompensa acumulada. Primer ampliem el treball existent en proposar un model que no sols resol conflictes, sinó que també mostra que és possible considerar diversos factors que afecten l'eficiència i l'impacte ambiental. Aquest enfocament és reforçat per l'ús de Graph Neural Networks (GNN), que faciliten la cooperació i la comunicació. Mostrem que a través de la cooperació és possible que els agents autònoms aprenguin estratègies de resolució que són semblants a les estratègies conegudes pels controladors humans. Aquests resultats no havien estat reportats a la literatura de GNN.

Si bé aquests dos objectius poden semblar separats, en aquesta tesi argumentem que un sistema unificat de gestió de la complexitat del trànsit aeri amb aplicacions de la IA pot contribuir a avançar més en els sistemes autònoms al sector aeronàutic. Aquesta tesi conclou amb una visió de la IA a l'aviació que considera significatiu l'humà en el llaç de control i l'alineació de valors com a essència de les línies de recerca futura.

PUBLISHED CONTENT AND CONTRIBUTIONS

Journal and Conference Articles

- Isufaj R, Koca T, Piera MA. Spatiotemporal Graph Indicators for Air Traffic Complexity Analysis. Aerospace. 2021; 8(12):364. DOI: https://doi.org/10.3390/aerospace8120364
- Isufaj R, Omeri M, Piera MA. Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning. Applied Sciences. 2022; 12(2):610. DOI: https://doi.org/10.3390/app12020610
- Isufaj, R., Aranega Sebastia, D., Angel Piera, M. (2022). Toward Conflict Resolution with Deep Multi-Agent Reinforcement Learning. Journal of Air Transportation, 1-10. DOI: https://doi.org/10.2514/1.D0296 (*This article was originally submitted to ATM Seminar 2021, where it won the best paper award for the Separation track. As a result, an extended version was published in Journal of Air Transportation*)
- Isufaj, R., Omeri, M., Piera, M. A., Valls, J. S., Gallego, C. E. V. (2022). From Single Aircraft to Communities: A Neutral Interpretation of Air Traffic Complexity Dynamics. arXiv preprint arXiv:2208.01740. (*Currently under review*)
- Omeri, M., Isufaj, R., Ortiz, R. M. (2022). Quantifying Well Clear for autonomous small UAS. IEEE Access, 10, 68365-68383. DOI: 10.1109/ACCESS.2022.3186025
- Koca, T., Isufaj, R., Piera, M. A. (2019). Strategies to Mitigate Tight Spatial Bounds Between Conflicts in Dense Traffic Situations. Proceedings of the 9th SESAR Innovation Days, Athens, Greece, 2-5.
- Kola I., Isufaj R., Jonker C. (2022) Does Personalization Help? Predicting How Social Situations Affect Personal Values. Proceedings of the First International Conference on Hybrid Human-Machine Intelligence

Supervised Theses

- David Aranega Sebastia - Conflict Resolution with Deep Multi-Agent Reinforcement Learning - *Master Thesis*

- Javier Garcia Cañadillas - Analysing Air Traffic Complexity in High Density sUAS Operations - *Master Thesis* (Co-supervised with Marsel Omeri)
- Jaume Saez i Valls - Modelado y Simulación De La Complejidad del Tráfico Aéreo - *Bachelor Thesis*
- Lichen Xia - Opponent Modeling in Automated Negotiation Applied to P2P Energy Trading - *Master Thesis* (Co-supervised with Dr. Luciano Cavalcante Siebert at TU Delft)

Summer Schools and Workshops

- 1st Engage KTN summer school, September 2019, Belgrade, Serbia
- 2nd Engage KTN workshop on Data-driven Trajectory Prediction, December 2019, Athens, Greece
- 2nd Engage KTN summer school, September 2020, held online
- 3rd Engage KTN summer school, September 2021, held online

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Air Traffic Management

In the early days of aviation, the density of traffic was low enough such that the pilot alone was responsible for the safety of the aircraft. They needed to take measures in order to avoid other aircraft, terrain or the ground. However, the tremendous increase in traffic demand has resulted in the captain gradually losing the ability to carry out all maneuvers required for a safe flight [1]. These developments lead to the creation of Air Traffic Control (ATC). Little progress was made on ATC until after World War II, when governments were forced to take actions to ensure the safety and efficiency of flights. According to Cook [1], this lead to better route structures, more advanced and efficient radios and navigation aids and to the establishment of procedures and standards that most national administrations subscribed to by the International Civil Aviation Organisation (ICAO).

ATM Structure

Nowadays, air transportation has grown into a complex system comprising of numerous actors encompassing different aspects of air traffic. Broadly speaking, there are five families of services that allow for air transportation to be possible: Communication, Navigation and Surveillance (CNS), Search and Rescue (SAR), Aeronautical Information Service (AIS), and Meteorological services for air navigation (MET) and Air Traffic Management (ATM) [2], which is also the focus of the work in this thesis.

ATM is formally defined as the dynamic, integrated management of air traffic and airspace safely, economically and efficiently, through the provision of facilities and seamless services in collaboration with all parties and involving airborne and ground-based functions [3]. Such a system implies a complex socio-technical systems, which at present is comprised by three main layers: air space management (ASM), air traffic flow management (ATFM) and air traffic services (ATS), as shown in Figure 1.1

ASM determines the structure of the airspace, while ATFM groups en-route aircraft into flows. It is worth noting that in Europe, ATFM has expanded into air traffic

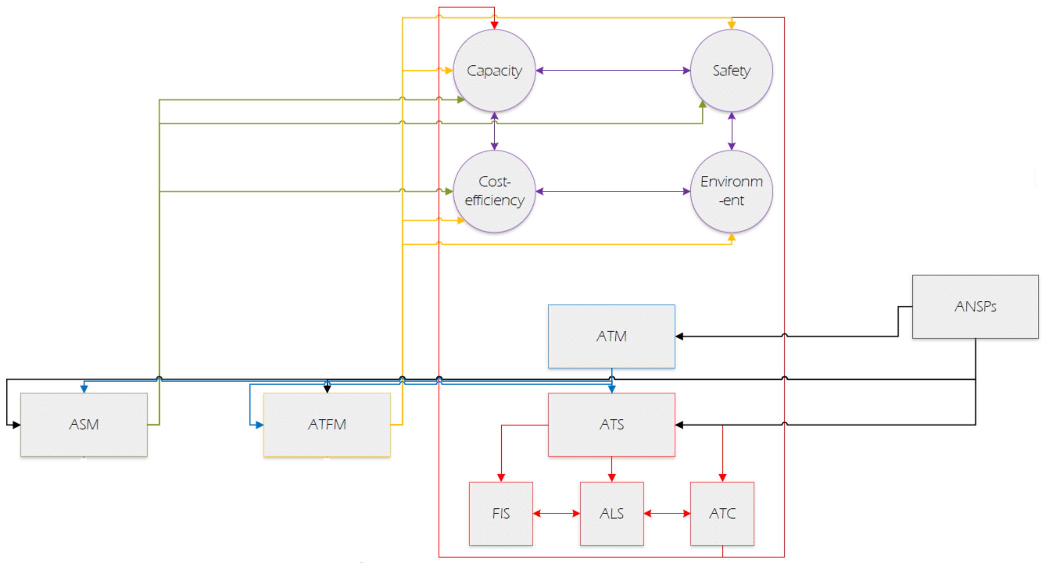


Figure 1.1: Organisation of ATM into several several layers. The full figure with all components and inter-relations between them can be found in Chapter 3.

flow and capacity management (ATFCM) [4]. The purpose of ATFCM is to avoid the overloading of parts of the airspace (same as ATFM), while ensuring that the capacity is fully exploited, which is crucial in dense and busy airspace.

ATS is entrusted with providing decision making and advisory services and is itself comprised of several parts: alerting service (ALS), flight information service (FIS) and air traffic controle service (ATC). As the name suggests, ALS must notify the appropriate stakeholders regarding aircraft in need of search and rescue. On the other hand, FIS' purpose is to give advice and information for the safe and efficient conduct of flights. Finally, ATC must prevent separation minima infringements and facilitate an orderly flow of air traffic.

Performance of ATM layers

While all these layers are part of the ATM structure, they have different objectives, which in practice compete to maximize their own goals. The performance of these layers of ATM is measured through various Key Performance Areas (KPAs), where some of the most important are safety, capacity, cost-efficiency and environment [5], shown in Figure 1.1.

The relationship and interdependencies between these KPAs have been subject of study for several SESAR projects such as STREAM [6] and APACHE [7]. Nevertheless, these projects claim that further research is necessary to fully understand these interdependencies. Consequently, the aeronautic community has accepted the term *emergent dynamics*, which comes as a result of the un-modelled behavioural dynamics between the aforementioned KPAs.

Air Traffic Complexity

Air traffic complexity has been a prominent research topic since the early days of ATC operations. However, there still is no definitive answer to what constitutes a complex situation in the context of ATM. Initially, the majority of the work related air traffic complexity with the workload of controllers, but this proved to be very difficult, due to the subjective nature of this measure. More recent work has focused on providing more objective indicators for air traffic complexity that are based on purely the geometry of the traffic, or combine the geometrical approach with the more subjective complexity scores.

Complexity of air traffic is one of the key topics of this work, and we show how it can be used to smoothen the different Key Performance Indicators (KPIs) that comprise the KPAs elaborated in the previous subsection. A more thorough overview of existing work on complexity can be found in Chapters 2,3 and 4.

1.2 Conflict Management

A large portion of this work is focused on Conflict Detection & Resolution (CD & R), thus it is worthwhile to give an overview of conflict management in ATM.

According to ICAO Doc9854 [8], the goal of conflict management is to limit, to an acceptable level, the rist of collision between aircraft and hazard, where a hazard can be: other aircraft, terrain, weather, wake turbulence, incompatible airspace activity and surface vehicles and other obstructions when the aircraft is on the ground. As defined by ICAO, conflict management consists of Strategic Conflict Management, Tactical Conflict Management and Collision Avoidance, as shown in Figure 1.2.

Strategic Conflict Management

As specified by ICAO Doc9854 [8], the first layer of conflict management is Strategic Conflict Management. This is realized through airspace organization and management, demand and capacity balancing and traffic synchronisation components. In this first layer, the majority of actions will occur prior to departure. However, there are cases that strategic actions might be required after take-off, particularly in long duration flights. The goal of the strategic layer is to reduce the need to apply the second layer (next section) to an appropriate level as determined by the ATM system

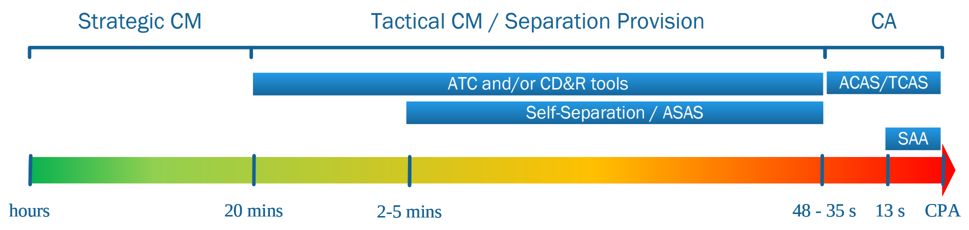


Figure 1.2: Time frames of ATM conflict management. Taken from Omeri et al.[9]

design and operation.

Tactical Conflict Management

At the tactical level, conflict management is performed by ATC, which provides guidance and information to the pilots through Air Traffic Control Operators (ATCo). ATCos are responsible for *sectors* which are controlled airspace volumes. In Europe, each sector has a pair of ATCos: the executive controller and the planner [1]. Some of the responsabilities of the planner are to coordinate and approve the entry and exit of flights into the sector, identify flight paths which are least likely to generate conflicts and monitor additional frequencies. On the other hand, the executive controller is responsible for communicating with the pilots, accepting aircraft into the sector, monitoring the progress of aircraft and attaining separation management.

There has been considerable effort in the community to develop advanced automated tools to assist controllers in their duties. One of such areas of interest has been the development of CD & R tools. A framework for CD & R *decision support tools* has been introduced in [10]. They argue that such a support tool should have three fundamental processes: detect the conflict, communicate the detected conflict to the ATCos, assist in the resolution of the conflict.

The airspace is monitored and appropriate state information is gathered by various sensors. The state must provide an estimate of the current traffic situation. It is important to consider also uncertainties in the state estimation as a result of the type of sensors used, or sensor errors. The trajectories are projected into the future as per some dynamic trajectory model. These predictions are combined with some metrics (e.g., minimum pairwise separation) in order to determine if the ATCo needs to be informed or not. Moreover, resolutions must be generated by the tool and then presented to the controller. However, in the case of fully automated conflict resolution, the (now autonomous) system must implement the resolution.

Collision Avoidance

In the event of an emerging collision, the Collision Avoidance System is activated seconds before the closes point of approach (CPA). Standard CA systems for most commercial aircraft are Traffic Collision Avoidance System (TCAS) and Airborne Collision Avoidance System (ACAS) [11] and Airborne Collision Avoidance System (ACAS) [12]. The goal of these systems is pairwise (or multi-threat [13]) collision avoidance and factors that affect their performance are encounter geometry, closure rate and flight level. Finally, the principle of See and Avoid also serves as a CA method, specifically for operations in uncontrolled airspace and in general cases where aircraft are not equipped with TCAS or similar systems.

1.3 UAS Traffic Management

Part of the work in this thesis has been applied in the context of safety for unmanned aircraft systems (UAS). The expected increase of commercial and civil applications of UAS will be followed by increased density and new challenges mainly related to safety, reliability and efficiency of airspace management. Thus, crucial to the deployment of UAS operations with be the development and implementation of a UAS Traffic Management (UTM) systems. Several system architectures and models have been developed in the last few years. Among them, we distinguish NASA-UTM developed by the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA) in the USA and U-Space in Europe where Single European Sky ATM Research Joint Undertaking (SESAR JU) is leading the efforts[1].

Similar to manned aviation, a conflict refers to a state or situation where a predetermined separation minima has been compromised by two or more aircraft [8]. The event when a separation is infringed is referred to as a loss of separation (LOS). To prevent LoS, UAS make use of separation provision, which is the tactical process for keeping aircraft away from hazards, which could be implemented as a UTM service.

The self-separation function is carried by the Detect and Avoid (DAA) system of the UAS, which is intended to comply with regulatory requirements to remain *well clear* of other airborne hazards [14]. A nominal DAA should comprise of three

[1]We note that the work in this thesis is not specific to any of them

main modules: conflict detection, alerting and conflict resolution.

Conflict Management

In this thesis, the UAS conflict management framework is the one proposed by Omeri et al.[2] [9], which aligns with SESAR/NASA-UTM concepts [15, 16].

This framework follows a similar conflict management structure as ATM: strategic conflict management, tactical conflict management (i.e., separation provision) and collision avoidance. It spans four stages that assess the aforementioned safety layers.

Stage 1 - Strategic Conflict Mitigation: conflicts are detected and resolved before take-off, based on their flight plans submitted to the UTM. It involves removing intersecting trajectories on spatio-temporal basis and then re-planning to satisfy various constraints such as no-fly zones, weather or other obstacles

Stage 2 - Separation Provision Service: Similar to ATC duties, UTM must offer as a service for in-flight separation, in case that flight plans are still not conflict-free[3]. The aircraft that are subscribed to this service will get early awareness (e.g., through alarms) for possible LoS between other aircraft, which may be manned or unmanned, and guidance for safe and efficient resolutions.

Stage 3 - Self-Separation: Self-Separation is derived from the concept of Free Flight [17] and relies on the capabilities of the aircraft to maintain a safe separation minima from other airspace users (AUs). This functionality could be carried out by the remote pilot (i.e., manually), assisted or fully automated. It removes the responsibility from the UTM and delegates it to the aircraft.

Stage 4 - Collision Avoidance: provides a final safety layer in order to prevent mid-air collisions. It is characterized by imminent and sharp maneuvers [4]. Similarly to Stage 3, it can be managed by the pilot or autonomously.

The framework for managing UAS conflicts is illustrated in Figure 1.3. Stage 2 is seen as a ground based service, requiring reliable communication between the UAS and the UTM. In cases when the encounter with the mid-air intruder is not resolved by Stage 2, the aircraft must use DAA capabilities to resolve the conflict. A typical DAA is composed of CNS subsystems, sensors, conflict detection module, alerting and guidance algorithms, ground control station and command and control (C2C) subsystems.

[2]We note that this is joint work conducted in the context of this thesis

[3]After the Stage 1 service has been invoked

[4]Could also transition to a hovering state, depending on the capabilities of the aircraft

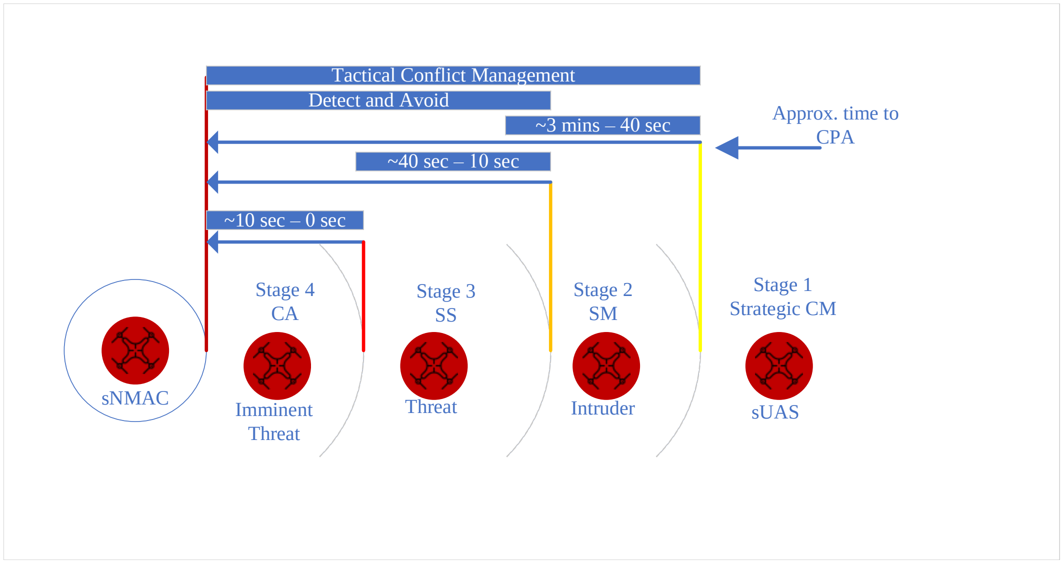


Figure 1.3: UAS conflict management framework. Taken from Omeri et al.[9]

In case of autonomous flights, the navigation and maneuvers are made possible by the use of a flight computer, referred to as the autopilot (AP).

1.4 A Brief Introduction to Artificial Intelligence

This section serves to give an overview of the field of Artificial Intelligence (AI). Owing to the structure of this thesis, the specific methods of AI utilized here will be formally introduced in subsequent chapters.

Knowledge Base AI

The idea of a "thinking machine" has long been part of human imagination. When programmable computers were first formulated, people wondered whether they might become intelligent [18]. AI is one of the newest fields in science and engineering. Today, AI is a thriving field with many practical applications and active research topics. Nevertheless, there is not one single definition of what AI is. In their book, Russell and Norvig [19] give a summary of several definitions, which they group along several dimensions. Some of the definitions are concerned with *thought processes* and *reasoning*, where others address *behavior*. Each of these perspectives has been followed through different approaches. For instance, a *human-centered* approach is in part empirical science, while a more *rationalist* approach involves a combination of mathematics and engineering.

During its infancy, AI was focused on tackling problems that were intellectually difficult for humans, but could be described by formal, mathematical rules. Many of the early successes happened in these sterile and formal settings, where the AI did was not required to have or gain much knowledge about the world it was operating in[5]. These early AI projects hard-coded the knowledge about the world in formal languages. The AI then reasoned about statements in these formal languages using logical inference rules. Such systems were known as *knowledge base* approaches to AI. However, the knowledge and facts required for such an approach to work had to be devised and entered into the model by human supervisors. Quickly it became clear that people struggled to accurately describe complex information about the world in the proposed formal languages and knowledge base AI failed to have any major breakthroughs [18, 20, 19].

Machine Learning

The shortcomings of AI models that relied on hard-coded knowledge indicated that such systems needed to have the ability to learn (i.e., acquire their own knowledge), through the extraction of patterns from raw data. Such a capacity is known as *machine learning* (ML) [18]. Machine learning algorithms improve their performance by making observations about the world[6]. There are three main reasons why an AI system should be able to learn [19]:

- When dealing with a complex problem, the designers of the AI might not be able to anticipate all scenarios the AI might find itself in.
- When the AI needs to make predictions for a certain problem, the designers might not be able to accurately make the prediction themselves
- The designers of the AI are unable to solve the problem they are tackling. A typical example includes creating a computer program that recognizes faces. The best performing programs are those that use learning algorithms.

Currently, there are three paradigms of machine learning algorithms, classified by the type of feedback that the AI receives [19]:

- **Unsupervised learning (UL)** The AI model must learn patterns in the input even though no explicit feedback is supplied.

[5]A typical example is IBM's Deep Blue chess system, which beat world champion Garry Kasparov

[6]Here observation can be through interacting with the environment or analysing historical data from the environment

- **Supervised learning (SL)** The AI model observes some input-output pairs and learns a function that maps the input to the output.
- **Reinforcement learning (RL)** The AI model learns through interacting with the environment. It receives a series of rewards which indicate how the model is performing

However, there are other paradigms, which arise in situations when the distinctions between the aforementioned paradigms are not clear. For instance, in semi-supervised learning, the model is given a few labeled (i.e., the output for the corresponding input) and it must learn what it can from a large number of unlabeled examples. Another layer of complexity is added when the labels are not always correct. As such, the lack and noise of labels create a hybrid between supervised and unsupervised learning.

The performance of machine learning algorithms largely depends on the representation of the data they are given. For instance, when a ML algorithm is tasked with determining the type of an aircraft, it does not observe the aircraft directly. Instead, it must be fed relevant pieces of information such as, the number of engines, wingspan etc. Such pieces of information are known as *features*. For instance, a simple logistic regression algorithm learns how each of the features of the aircraft correlates with the outcomes (i.e., type of aircraft). There is a large number of tasks that can be solved by ML algorithms by extracting the right set of features.

Deep Learning

Nevertheless, there are many tasks where it is difficult to know which features are useful. To illustrate, let us imagine writing a program that detects aircraft in a photograph. One could imagine that a feature could the presence of wings, however, it is not clear how to describe exactly what wings are in terms of pixel values. The solution to this problem is to use ML algorithms to not only create a mapping from input to output, but also to learn a meaningful representation of the input itself. This approach is known as representation learning. Such an algorithm is able to discover an appropriate of features from simple to complex tasks. ML algorithms based on this approach have proven to perform much better than traditional ML [18].

The goal of representation learning is to determine the "factors of variation" that explain the observed data [18]. In the previously used example of detecting aircraft in a photograph, such factors include the position of the aircraft, the brightness and angle of the sun. Many real-world applications of AI have many factors of variance that influence every piece of the observed data and thus most applications require discarding factors that are not relevant. However, the extraction of such abstract features from raw data can be very difficult and representation learning is not able to solve this problem.

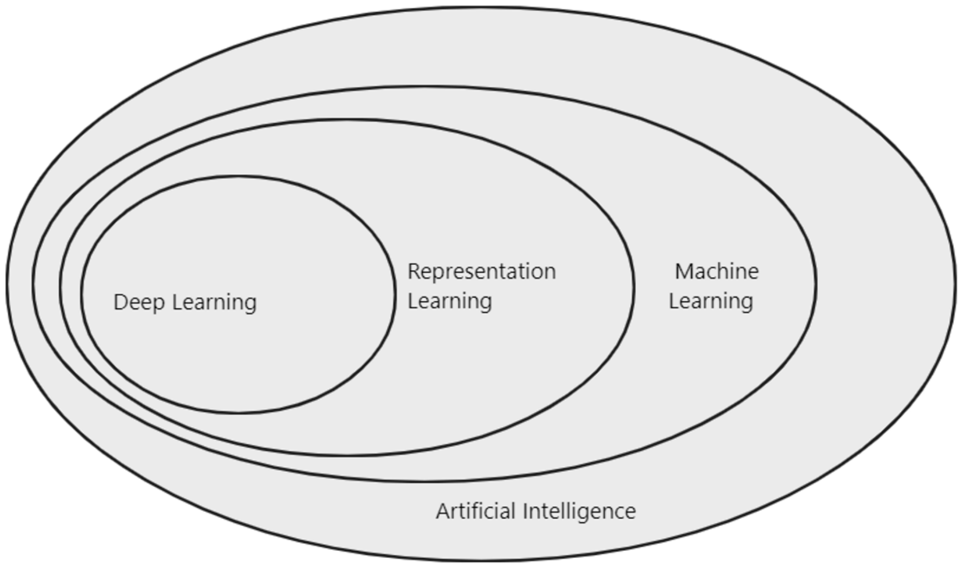


Figure 1.4: A Venn diagram showing the different disciplines of AI.

Deep learning (DL) is such an attempt to solve this problem, by introducing representations that are expressed in terms of simpler representations. That is, deep learning allows the AI model to build complex concepts out of simpler ones. The typical example of a deep learning model is the multilayer perceptron (MLP)[7]. An MLP is a mathematical function that maps input values to output values, but the function is composed by many simpler functions. Applications of these different functions provide new representations of the input. These methods discover intricate structures in large data sets by using the backpropagation algorithm (or variants of) to indicate how the machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer [21]. For instance, the use of convolutional networks has resulted in breakthroughs in image and video processing, recurrent neural networks have been used on sequential data such as text and speech, whereas graph neural networks have

[7] Also called a feedforward network

made possible to use conventional DL algorithms in non-Euclidean data such as social networks.

A summary of the different disciplines of AI is shown in Figure 1.4

1.5 Objectives of the Dissertation

The main research goal of this thesis has been to advance the state of the art in air traffic complexity and automated air traffic CR.

Both of these topics have been heavily investigated in the aeronautic community and have been subject to numerous research projects. Nevertheless, there are several issues that are not treated in the current literature. These issues form the objectives of this thesis and will be presented in detail:

- **Provide a generic definition for pairwise interdependencies**

  We define pairwise spatiotemporal interdependencies between aircraft by considering the (horizontal and vertical) distance at a time step. The interdependency is weighed between 0 and 1 and the evolution in a time window is considered (Chapter 3). This definition is generic in the sense is adaptive to the use-case and context. For instance, in Chapter 3 the interdependencies are specific to the sector, while in Appendix A we show how it can be adapted for the case of UAS.

- **Define air traffic complexity in such a way that provides elaborate information efficiently to different stakeholders**

  We use graph theory to model air traffic as a dynamic graph[8] and define four complexity indicators that combine different topological information and the severity of the interdependencies to give a detailed and nuanced evolution of complexity in a certain airspace (Chapter 3). The effectiveness of the proposed indicators in a different domain is shown by applying them to high density sUAS scenarios (Appendix A).

- **Develop a methodology that identifies high complexity spatiotemporal areas in a sector**

  The proposed indicators are extended to define the concept of *single aircraft complexity*. This concept is in turn used to define complex communities (Chapter 4).

[8] A dynamic graph is a graph that evolves through time

- **Formalize CR as an AI problem and investigate how learning affects the resolution of conflicts**

  CR is modelled as a Multi-Agent Reinforcement Learning (MARL) problem where conflicting aircraft are modelled as agents capable of decision making (Chapter 5).

- **Provide an AI model that goes beyond simply solving conflicts**

  A reward function is designed that considers the interests of different stakeholders and includes factors such as fuel consumption, time to LOS, complexity etc.

- **Investigate the scalability of MARL algorithms in the context of automated CR**

  MARL based on Graph Neural Networks (GNNs) has been used to develop a model that considers many conflicting agents (Chapter 6).

1.6 Overview

The remainder of this dissertation beyond Chapter 2, which provides a summary of the most relevant work regarding the topics of interest, is organized as a compendium of research articles written as part of the work in this thesis. Chapter 3 presents "Spatiotemporal Indicators for Air Traffic Complexity Analysis"[9] which provides the definition of air traffic complexity based on graph theory. In Chapter 4, we present "From Single Aircraft to Communities: A Neutral Interpretation of Air Traffic Complexity Dynamics"[10], where the concepts of single aircraft complexity and complex community are defined. Chapter 5 is "Towards Conflict Resolution with Multi-Agent Reinforcement Learning"[11] where we define CR as a MARL problem. Furthermore, a reward function is designed that considers the interests of different stakeholders. In Chapter 7, we show "Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning"[12] which models CR as a MARL problem based on GNNs in order to tackle scalability. Finally, Chapter 7 contains a detailed account of the contributions of this thesis and future work based on the developed ideas.

[9]published in MDPI Aerospace

[10]submitted to MDPI Aerospace

[11]This article was originally submitted to ATM Seminar 2021, where it won the best paper award for the Separation track. As a result, an extended version was published in Journal of Air Transportation, which is shown here

[12]published in MDPI Applied Sciences

Chapter 2

STATE OF THE ART

Due to the structure of this thesis, relevant related work will be discussed in each chapter. Nevertheless, in this chapter we will summarize results from three research directions pertinent to our work.

2.1 Air Traffic Complexity

Complexity as a measure of ATCo workload

As previously mentioned, air traffic has been a common research topic in the ATM community, with the first papers on the topic written in the 1960s [22]. Since then, the majority of works have related air traffic complexity with its influence on controllers [23, 24], where two basic complexity factors were identified: sector complexity and traffic complexity. Further work [25] laid out the hypothesis that complexity causes the cognitive workload of controllers to change.

The most common metric in this group is Aircraft Density [26, 27], defined as the number of aircraft at the time of measurement, however as other work has suggested [28], this measure only weakly correlates to the workload.

Schmidt [29] introduced the control difficulty process, which is calculated sum of the expected frequency of occurrence of events that affect the controller workload. Each event is weighted by the amount of time needed to deal with it.

Dynamic Density (DD) [30, 31] was introduced in order to measure the workload of ATCos by considering traffic density and several factors that were supposed to contribute to traffic complexity (e.g., number of ascending or descending aircraft). DD is defined as a weighted sum of these attributes, which makes it suitable for predicting future workload. Interval Complexity [32], is similar to DD but is calculated as the average over a 5–10 min window.

Chaboud et al. [33] investigated the influence of complexity on workload and services costs, while Flynn et al. [34] studied complexity in the context of sector categorisation. The relationship between complexity and workload remains actual [35, 36] with more recent works focusing on measuring the cognitive workload of controllers directly. They include methods such as measuring ocular activity [37], brain imaging [38] and electroencephalography [39].

Andraši et al. [40] propose a model based on ML that can be used to determine subjective traffic complexity with a similar accuracy to linear models. Human-in-the-loop experiments were conducted with air traffic controllers which concluded that the variance in the subjective complexity scores cannot be explained by traffic characteristics. Furthermore they state that inconsistent ATCo ratings were one of the sources of errors.

Objective Methods for Complexity

Another group of works define complexity only on objective factors such as structure of air traffic. Delahaye et al. [41] propose two distinct ways of measuring air traffic complexity. Firstly, they consider geometrical properties in order to build a new complexity coordinate system in which sector complexity through time is presented. Secondly, they represent air traffic as a dynamic system in order to yield an intrinsic measure of complexity through Kolmogorov entropy.

Another metric that measures complexity through geometrical features is Fractal Dimension [42]. Complexity is calculated through the degrees of freedom of an aircraft given its route and constraints. Nonetheless, it comes with strong dependencies on airspace structure, requiring it to be made up of piece-wise linear segments.

The Conflict Activity Level (CAL) was introduced by Wee et al. [43] as a dynamic tactical complexity model. CAL evaluates the likely aircraft flight shape profile based on their current and projected position and trajectory, through which the complexity values are calculated. The authors report some level of consistency with existing complexity metrics in terms of ranking air traffic scenarios.

Several methods based on ML have also been proposed. Work done by Gianazza and colleagues [44, 45, 46] uses tree search and neural networks for predicting complexity. Their methods share the assumption that complexity in historical flight data increased prior to the splitting of the collpased sector into two smaller ones and decreased prior to collapsing the sectors into a larger one. The neural network could then predict the future increase in complexity. Tree search was ultimately used to determine the airspace configuration which yielded the lowest complexity for the given air traffic scenario.

Wang et al. [47] propose describing air traffic situations using the theory of complex networks. They provide a complexity vector, which is comprised from several indicators. Situations are then classified as 'low-complexity', 'medium-complexity' and 'high-complexity' depending on vector values. Their approach bears some similarities to the one of this thesis; however, as we will explain in Chapter 3, there are some key differences.

The majority of previous work on complexity have produced metrics that generally lack elaborated information for a proper understanding in a multi-stakeholder context. Furthermore, existing metrics in large reduce complexity into one score which loses some of the granularity of the information. Lastly, existing metrics do not investigate the implications of detailed information in the level of single aircraft.

2.2 Methodologies for Conflict Resolution

Similar to air traffic complexity, CR has also been an important topic of research in the community. There are several works that give an excellent overview of the existing research [48, 10, 49] based on more traditional approaches such as Mixed-Integer Nonlinear Programming (MINP), Monte Carlo simulations, spatiotemporal regions and Voltage Potential methodology. However, in this chapter we will summarize some works that focus on CR (for manned and unmanned aviation) using (single or multi agent) RL.

The earliest work we could find was that of Yang et al. [50], which used a heuristic based Q-algorithm for CR in UAS. This work was not based on Deep Reinforcement Learning (DRL).

The work of Ribeiro et al. [51] uses DRL to improve the resolutions given by the Modified Voltage Potential (MVP) algorithm. The work of Wen et al. [52] uses the same algorithm as that of [51], however it directly generates heading changes using a continuous action space. On the other hand, Hermans [53], trains a DRL model to solve conflicts based on demonstrations by human controllers. This can be considered as Inverse Reinforcement Learning (IRL).

Several works have been published by Brittain and Wei [54, 55, 56]. They tackle CR from different points of view using single and multi agent settings, which has resulted in several algorithms that consider centralized and decentralized architectures.

Pham et al. [57] train a single agent RL model which operates under uncertainties in actions. This means that when an agent takes an action, there is some uncertainty in which state it will end up.

Wang et al. [58] propose K-Control Actor-Critic in order to guide an intruder to make a fixed number of maneuvers for CR in the free flight scenario. Zhao et al. [59] introduce a model which integrated prior physics understanding and generates human-explainable results for decision making.

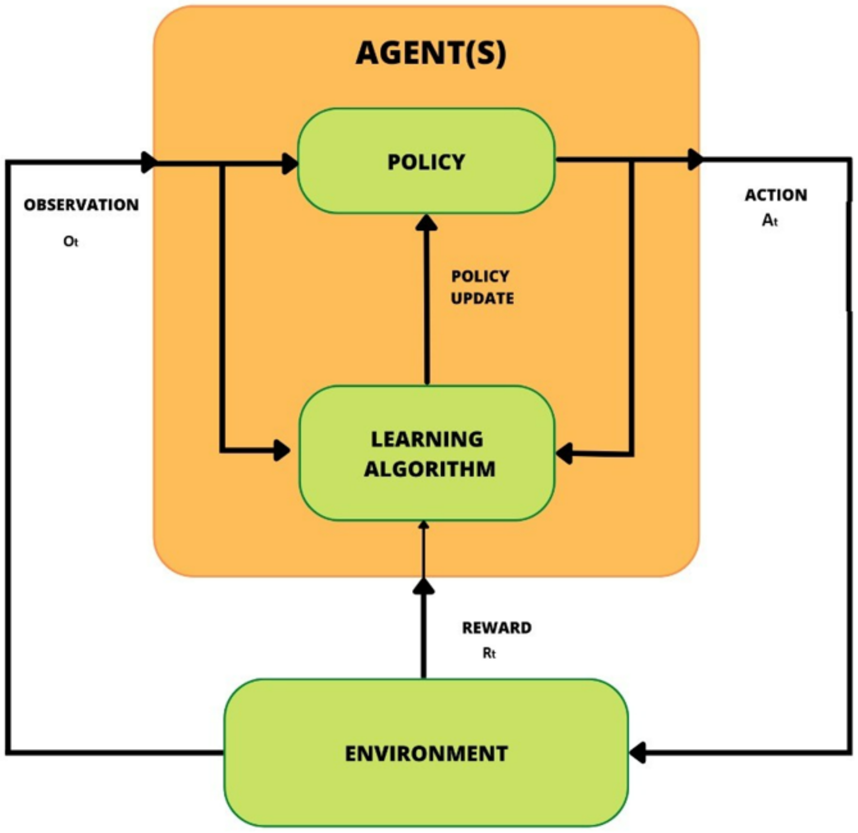


Figure 2.1: Training process for a single or multi agent generic RL problem.

DRL has also been used for CA, such as the work of Li et al. [60], which was shown to operate more efficiently than traditional methods in dense airspace while maintaining satisfactory levels of safety.

The use of GNNs has also been observed in some works. For instance, Mollinga et al. [61] propose a model that guides an arbitrary number of aircraft across unstructured airspace, which is similar to the work of Dalmau et al. [62], which envision a tool to support ATCos in complex traffic scenarios.

Through DRL, agents improve their abilities by interacting with the environment. This process, shown in Figure 2.1, is the same for all of the previously mentioned methods, including the work to be presented in the following chapters. Training is composed of training steps. In each step, the environment is in a certain state (e.g., aircraft positions, velocities etc). This state is observed by the agent(s)[1] through some mechanism, which in the simplest case could be sensors. The agent then generates an action according to its policy, which results in the environment to transition to another state. In turn, the environment sends to the agent a scalar reward, which indicates the agent how well they are doing in that state. Finally, the agents updates its policy according to the learning algorithm, and the whole process repeats. Ideally, the trained model would be then validated through some rigorous procedure to ensure the quality of the resolutions.

The majority of existing work on CR with RL takes a traditional view, in the sense that the main goal of the model is to simply solve conflicts. However, in this thesis we show that it is possible (and in fact necessary) to consider several other factors that affect efficiency and the environmental impact. Furthermore, we show that through cooperation it is possible for autonomous agents to learn resolution strategies that are similar to known strategies by human controllers, which existing work using GNNs has not been able to achieve.

Other Applications of AI in ATM

Recently, there have a been number of projects supported by SESAR that have focused on applying ML/DL in various areas of ATM. These applications have been applied to better understand the underlying patterns of traffic and ATCo instructions.

The AI Situational Awareness Foundation for Advancing Automation (AISA) project (e.g. [63]) investigated how to increase automation in air traffic management. They aimed to explore the effects of human-machine distributed situational awareness and opportunities for automation of monitoring tasks in en-route operation.

The INTUIT project (e.g. [64] developed visual analytics and ML techniques to understand the trade-offs between key performance areas[2].

Moreover, DART (e.g. [65]), tackled the problem of trajectory prediction based on ML, in order to esitmate aircraft performance before or during the flight. The models were trained on previously recorded trajectories.

On the other hand The BigData4ATM project (e.g. [66]) investigated how different passenger-centri geo-located data can be anaylsed in order to identify patterns in passenger behavior, choice of travel of mode etc. The overarching goal of this

[1]Depending if the setting is with single or multiple agents

[2]This is a topic tackled in this thesis, in Chapter 4

project was to enable optimised decision making for the benefit of passengers and goods.

The MALORCA project (e.g. [67]) investigated speech application for various uses at an airport. For instance, ATCos give most of their instructions to pilots via voice communications. This requires that controllers make manual inputs to keep the system data correct. MALORCA used automatic speech recognition by converting speech to text for input into the system.

Finally, the E-Pilots (e.g. [68]) proposed a cockpit-deployable machine learning system to support flight crew go-around decision-making based on the prediction of a hard landing event.

Chapter 3

SPATIOTEMPORAL INDICATORS FOR AIR TRAFFIC COMPLEXITY ANALYSIS

There has been extensive research in formalising air traffic complexity, but existing works focus mainly on a metric to tie down the peak air traffic controllers workload rather than a dynamic approach to complexity that could guide both strategical, pre-tactical and tactical actions for a smooth flow of aircraft. In this paper, aircraft interdependencies are formalized using graph theory and four complexity indicators are described, which combine spatio-temporal topological information with the severity of the interdependencies. These indicators can be used to predict the dynamic evolution of complexity, by not giving one single score, but measuring complexity in a time window. Results show that these indicators can capture complex spatio-temporal areas in a sector and give a detailed and nuanced view of sector complexity.

3.1 Introduction

The mission of air traffic management (ATM) is to make air traffic possible by means of efficient, environmentally friendly and socially valuable systems [69, 70]. 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 biggest responsibility of ATCos is guaranteeing safety, which means they must issue instructions to pilots, monitor traffic to maintain safety distances and so on. The ability of controllers to effectively fulfill these duties is constrained by their workload, which can be defined as the mental and physical work done by controllers to manage traffic [71]. Therefore, it is important to keep ATCo workload at acceptable levels.

Controller workload is not easy to predict or estimate and is related to various factors which can be qualitative and quantitative [41, 72]. However, several studies [72, 73, 31] state that airspace complexity accounts for a large portion of workload. Complexity has been a prominent topic of research in ATM (see Section 3.2 for a more detailed review). A predictor of complexity is key not only to support ATCos, but also for a more sustainable and efficient air traffic management system in which complexity effects could be mitigated at early stages. However, there are several drawbacks to already existing metrics. First of all, the majority of metrics do not give a detailed view of complexity. For instance, the most widely used metric is **Aircraft Density** [26, 27], which measures the number of aircraft flying in a sector. This metric however is shown not to be adequate in capturing ATCo workload. For instance, Delahaye and Puechmorel [41] take the example of sector capacity, which is defined as the maximum number of aircraft that can be accommodated in a given time period [74]. They observe cases where controllers accept more than the actual capacity and cases where they refuse traffic even if capacity has not been reached.

Other methods attempt to measure ATCo workload directly [37, 38, 39]. These methods, however, tend to be intrusive to the workflow of controllers, in addition to having huge computational costs, which make them not suitable for practical uses. The majority of these metrics do not consider how complexity evolves in time.

Another area where complexity metrics can impact ATCo workload and as a consequence ATM capacity, are conflict detection and resolution (CD & R) [75, 76, 77, 78] decision support tools. A desirable characteristic of such solvers will be the quality of solutions in terms of complexity, which means that a conflict resolution that leaves the sector in a more complex state should be discouraged in favor of solutions that ideally lead to lower complexity, while preserving safety.

In this work, air traffic is modelled through graph theory and complexity is defined as the connectivity of the graph. Four indicators are proposed that represent different aspects aircraft interaction. The indicators measure several structural properties of the (traffic as a) graph, thus giving different insights into complexity. *Edge density* measures the size of the graph with respect to a fully connected graph with maximal weights, which identifies the aircraft that will create interdependencies. *Strength* measures the severity of the interdependencies, as aircraft that are closer present a more complex situation. The *Clustering Coefficient* provides information regarding the neighbourhood of each aircraft and the *Nearest Neighbour Degree* identifies if interdependent aircraft point to neighbours that may also have various interdependencies. As results show, each of the indicators is necessary and the information gathered from them can capture different shapes of interdependencies. This is an important step towards tackling hotspots, which are complex spatio-temporal areas in a sector, rather than simply solving conflicts. Furthermore, the evolution of complexity through time is considered, which gives a detailed view of how aircraft interactions change in time.

The rest of this paper is organizes as follows: in Section 3.2, we elaborate on existing complexity metrics and discuss some of their drawbacks. The theoretical background and modeling of air traffic using graph theory is presented in Section 3.3. In section 3.4, we present the proposed complexity indicators. We evaluate the indicators in illustrative examples in section 3.5, while results obtained from real traffic are presented and discussed in Section 3.6. In section 3.7, we draw conclusions and propose steps for further research.

3.2 Related Work

First of all, there have been previous works that model air traffic as a graph. Koca et al. [79] effectively transform traffic to a graph representation by identifying relevant aircraft to a conflict by means of spatio-temporal regions. However, their work is applicable only to conflict resolution. Furthermore, their graph analysis requires the presence of pairwise conflicts and is not intended to provide a dynamic complexity metric that considers traffic evolution.

There are many different ATM complexity metrics used in literature. As is the case with this paper, the bulk of them focus on measuring sector complexity at the tactical level.

The complexity of a certain unit of airspace has been usually linked to controller workload [73]. This seems quite intuitive, the more work a controller has to do, the more complex the situation is. However, measuring it is not trivial. Workload is highly subjective and there is no definite consensus as to what constitutes it [37]. Research on complexity metrics can be generally put into two groups: metrics that correlate workload with certain physical attributes of the airspace and metrics that attempt to measure directly the workload on the controller.

The most common metric from the first group is **Aircraft Density** [26, 27], defined as the number of aircraft at the time of measurement. Another common metric is **Dynamic Density** (DD) [80, 30, 24]. There are several ways DD has been defined in literature, but the underlying core idea is to define complexity as the weighted sum of several attributes. Examples of such attributes include the number of aircraft, number of cruising, ascending and descending aircraft, speed, heading change, number and time to conflicts etc. **Interval Complexity** [32], is similar to DD but is calculated as the average over a 5-10 minute window. However, they do not give an evolution of complexity in time.

Another metric that measures complexity through geometrical features is **Fractal Dimension** [42]. Complexity is calculated through the degrees of freedom of an aircraft given its route and constraints. Nonetheless, it comes with strong dependencies on airspace structure, requiring it to be made up of piece-wise linear segments.

Delahaye et al. [41]. propose two distinct ways of measuring air traffic complexity. Firstly, they consider geometrical properties in order to build a new complexity coordinate system in which sector complexity through time is presented. Secondly, they represent air traffic as a dynamic system in order to yield an intrinsic measure of complexity through Kolmogorov entropy.

Wang et al. [47] propose describing air traffic situations using the theory of complex networks. They provide a complexity vector, which is comprised from several indicators. Situations are then classified as 'low-complexity', 'medium-complexity' and 'high-complexity' depending on vector values.

Their approach bears some similarities to this paper, however, as we will explain, there are some key differences. First of all, they consider two aircraft interdependent only if there is a potential conflict between them. While conflicts affect the complexity of a situation, they are not the only source of it [79, 41, 26, 31]. Furthermore, they do not consider the severity of interdependencies. In this work, two aircraft are interdependent if they are close enough horizontally and vertically. Additionally, we consider the severity of interdependencies with a conflict having maximal severity (see Section 3.3 for a detailed description).

The second group of complexity metrics consists of approaches that measure the cognitive workload controllers experience. They include methods such as measuring ocular activity [37], brain imaging [38] and electroencephalography [39].

COTTON [81] is a project funded by SESAR JU that investigates many aspects that affect complexity (geometric and cognitive), which they call complexity generators. Different airspace configurations and time horizons are considered. They identify the most influencing generators and combine them. Furthermore, they consider uncertainty which they model through Bayesian networks.

Both groups of approaches have several drawbacks. Most of the metrics in the first group take a very simple point of view (e.g. aircraft density). While this means that those metrics can be computed quickly, it also means that the information they provide is not comprehensive. These metrics are not sufficiently expressive, as providing only a complexity score does not give the controllers a detailed reasoning why a situation is complex or not. The geometrical approach by Delahaye et al. overcomes these drawbacks, however it is not clear how to interpret the proposed coordinate system. Furthermore, their second approach based on dynamic system theory requires a calibration time and suffers from higher computational costs. Moreover, it is more suited for measuring flow complexity.

On the other hand, the second group of approaches tends to use methods that are quite intrusive to the workflow of the controller. Additionally, they measure workload on each controller individually, which will require an unknown calibration time to specific controllers. Finally, many of these approaches show complexity only for a fixed point in time. This is another drawback of such approaches, as the evolution of traffic through time is an important indicator of how complex a situation is.

Automation in ATM requires a better understanding of potential threats that can impact traffic and as such, complexity metrics should provide elaborated insights about the right mitigation measures to apply.

3.3 Air Traffic Modelled as a Graph

In this section, a formal definition of graphs is given and some basic attributes important to this paper are introduced [82].

Some Definitions of Graph Theory

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 G. Then e joins the 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 $\mathcal{T}$ 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 no 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:

$$s(i) = \sum_{j=1}^{N} w_{i,j} \tag{3.1}$$

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:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} a_{i,j} = \begin{cases} 1, \text{ if } (v_i, v_j) \in E \\ 0, \text{ otherwise} \end{cases} \tag{3.2}$$

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:

$$A_w = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} a_{i,j} = \begin{cases} w_{i,j}, \text{ if } (v_i, v_j) \in E \\ 0, \text{ otherwise} \end{cases} \tag{3.3}$$

In the weighted case, the entries for the edges present in the graph correspond to the weight $w_{i,j}$ of the edge.

In the case of an undirected graph, the adjacency matrix is symmetric, while for directed graphs this does not hold.

Modelling Aircraft Spatio-Temporal Interdependencies Using Graphs

In this section, we describe the model that formalizes spatio-temporal interdependencies of en-route traffic as a **weighted undirected** graph.

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 a 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:

$$wh_{i,j}(t) = \begin{cases} 1 \text{ if } dh_{i,j}(t) \leq H \\ 0 \text{ if } dh_{i,j}(t) \geq thresh_h \\ \frac{thresh_h - dh_{i,j}(t)}{thresh_h - min_h} \text{ otherwise} \end{cases} \tag{3.4}$$

$$wv_{i,j}(t) = \begin{cases} 1 \text{ if } dv_{i,j}(t) \leq V \\ 0 \text{ if } dv_{i,j}(t) \geq thresh_v \\ \frac{thresh_v - dv_{i,j}(t)}{thresh_v - min_v} \text{ otherwise} \end{cases} \tag{3.5}$$

$$w_{i,j}(t) = \begin{cases} \frac{wh_{i,j}(t) + wv_{i,j}(t)}{2} \text{ if } wh_{i,j}(t) > 0 \ \& \ wv_{i,j}(t) > 0 \\ 0 \text{ otherwise} \end{cases} \tag{3.6}$$

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.

3.4 Complexity Indicators

There is a wealth of research present on graph complexity[83, 84, 85, 86, 87, 88, 89, 90]. Many different definitions of complexity exist, depending on what aspects of graphs are being studied.

In this work, graph complexity and in turn sector complexity, is defined as the connectivity of the graph. Furthermore, by modeling 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 [89, 90] 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 neighbor degree**.

Edge Density

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:

$$ED(G,t) = \frac{\sum_{(i,j)\in E} w_{i,j}(t)}{A(V_t)}, A(V_t) = \frac{|V_t|(|V_t|-1)}{2} \tag{3.7}$$

$|V_t|$ 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.

Figure 3.1 shows an arbitrary sector. The only interdependecy exists between AC_3 and AC_4. However, as there are four aircraft in the sector, the potential number of interdependecies is six, therefore the ED score for this sector is $\frac{1}{6}$.

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

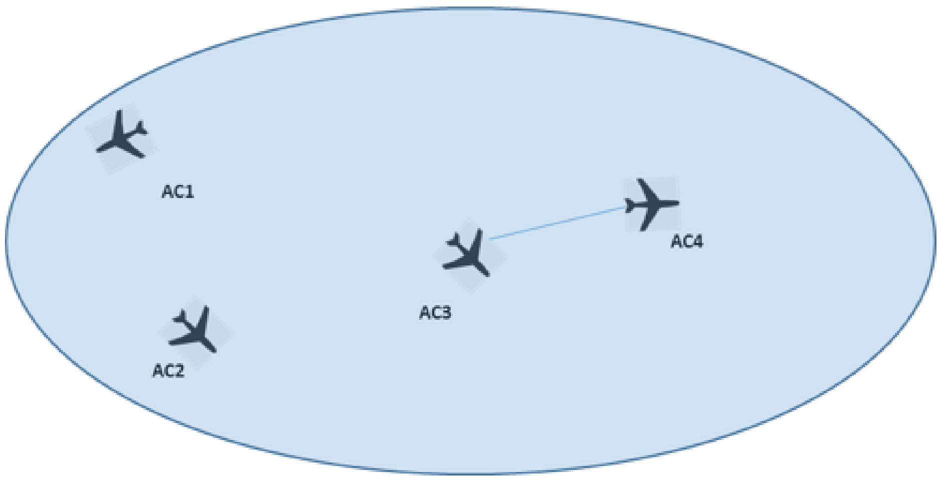


Figure 3.1: The edge density indicator.

aircraft in the graph. Formally, it is given as follows:

$$s(i,t) = \sum_{j=1}^{N} w_{i,j}(t) \tag{3.8}$$

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. This indicator is shown in Figure 3.2 where an arbitrary sector is shown in different time steps, There are three aircraft in the sector. AC_1 and AC_2 are moving closer while AC_1 and AC_3 are moving away from each other. As a result, the strength of AC3 decreases through time, however as AC_1 and AC_2 are getting closer, their strength score increases. With AC_1 having an increasingly stronger interdependency with AC_2, while its interdependency with AC_3 grows slightly weaker, strength increases in time for the whole sector. Strength can take values from 0 to $|V_t| - 1$, which happens in the case of a fully connected graphs with maximal weights.

Clustering coefficient

The clustering coefficient (CC) measures the local cohesiveness. This indicator provides information regarding the neighborhood of each vertex. It takes into account the weight of the clustered structure found in triplets. For each vertex i,

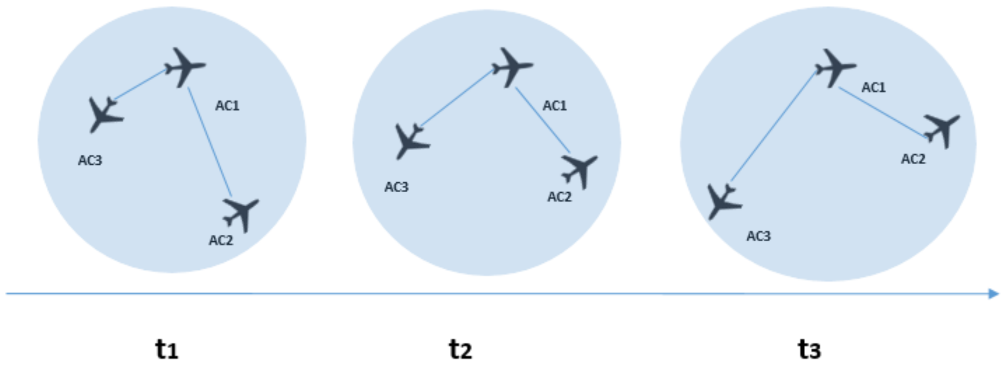


Figure 3.2: The strength indicator.

CC counts the number and the weight of triplets (see: Section 3.3) formed in the neighbourhood of i. Formally, the clustering coefficient of a vertex i, is calculated as follows:

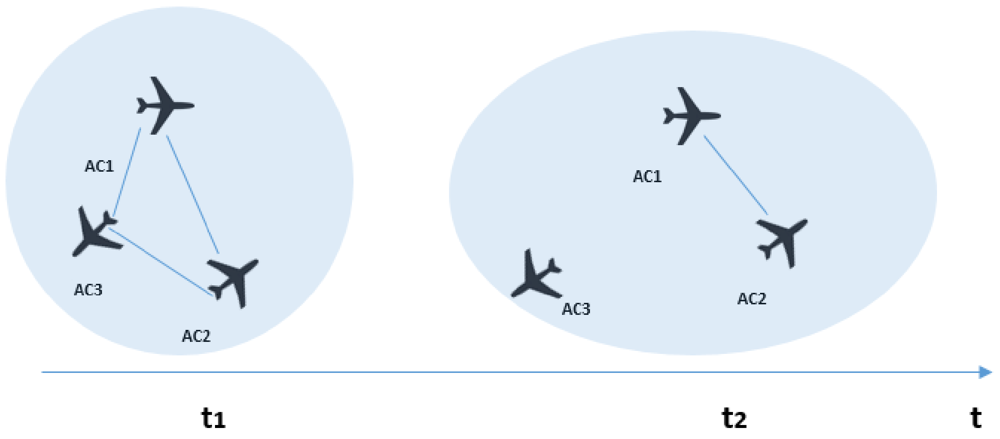


Figure 3.3: The CC indicator.

$$CC(i,t) = \frac{\sum_{j,k}(w_{i,j}(t) + w_{j,k}(t))}{2 \cdot (s(i,t)(deg(i,t) - 1)}, \forall (i,j,k) \in \mathcal{T}(t) \tag{3.9}$$

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.

The clustering coefficient indicator is illustrated in Figure 3.3. In t_1, the three aircraft form a cluster. Such a configuration of aircraft represents a situation where they are tightly connected. It can be observed that AC_3 is moving away from the two other aircraft, thus breaking the cluster. In the first timestep, the controller would need to be concerned with all of the aircraft present in the sector, while in the second timestep, an interdependency exists only between AC_1 and AC_2. This illustration shows how only measuring the strength is not enough to give a rich picture of complexity, but different topological characteristics need to be examined.

Nearest Neighbor Degree

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:

$$NND(i,t) = \frac{\sum_{j=1}^{N} w_{i,j}(t) deg(j,t)}{s(i,t)} \tag{3.10}$$

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. This indicator is illustrated in Figure 3.4.

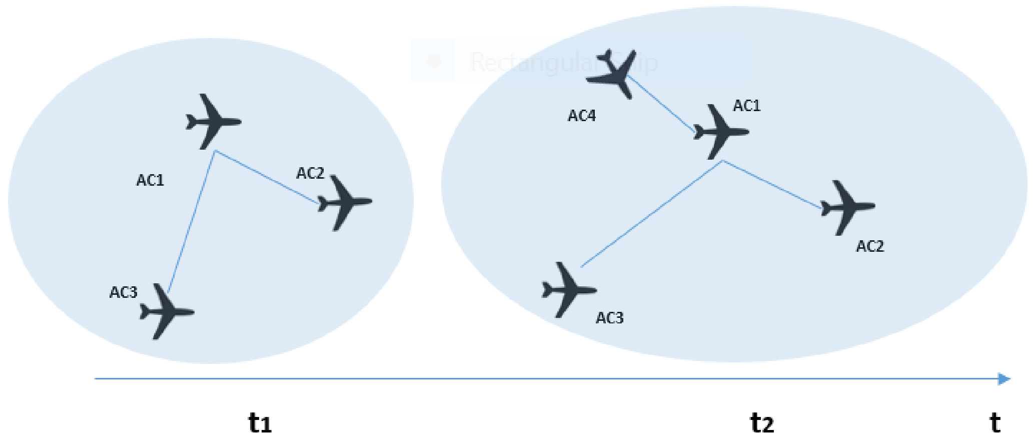


Figure 3.4: The NND indicator.

In t_2, AC_4 joins the sector, and immediately creates an interdependency with AC_1. The strength of AC_4, is therefore equal to its interdependecy with AC_1. This would indicate that its complexity is quite low, however AC_1 forms interdependencies with the rest of aircraft in the sector. As such, any potential maneuvers applied to AC_4 could affect AC_1, which in turn would affect the rest of the aircraft. Such a situation illustrates how interdependencies between aircraft where at least one has a high degree can lead to a highly complex situation. In this case, no clusters are formed, therefore the CC indicator does not change its value. This shows how clustering is not the only topological characteristic that captures the complexity of a situation, but all indicators are required to give a nuanced view of complexity.

Algorithm 4 shows the pseudocode of the procedure to calculate the complexity of a given airspace. The inputs are the airspace (e.g. coordinates of sector boundary), the time window for which to calculate the complexity, sampling time (e.g. measure complexity every 15 seconds) and the thresholds to determine interdependencies. The scores of the indicators are initialized as empty lists. Then, for each sampling time, until the duration of the time window has been reached, a snapshot of the traffic in the airspace is taken. After that, the graph is generated taking into account the threshold values for interdependencies. Then the indicator values for the current time are calculated using Equations 7,8,9 and 10 and the values for each indicator are returned.

3.5 Test Scenarios

In this section, we illustrate how the indicators behave by means of synthetic trajectories. After that, Miles-in-Trail scenarios and real traffic are used to verify the expected benefits.

Illustrative Examples

To provide a better understanding of the complexity indicators, three synthetic scenarios are shown.

Figure 3.5 shows a graph with relatively high connectivity, while Figure 3.6 shows a less connected graph. Both graphs have the same number of vertices and common edges have the same weight. G_1 has more edges.

Table 3.1 shows the indicator scores for G_1, while Table 3.2 shows them for G_2. As expected, because G_1 has more edges, it has a higher ED score.

In the case of CC, there are no clusters formed for G_2 Therefore, CC for each of

Algorithm 1 Calculate Complexity Algorithm

```
procedure COMPLEXITY($airspace, t_{window}, t_{sample}, threshold$)
    $ED \leftarrow \emptyset$
    $CC \leftarrow \emptyset$
    $NND \leftarrow \emptyset$
    $Strength \leftarrow \emptyset$
    $t_{curr} \leftarrow 0$
    while $t_{curr} <= t_{window}$ do
        $traffic_{t_{curr}} \leftarrow$GET-TRAFFIC($t_{curr}, airspace$)
        $graph \leftarrow$GENERATE-GRAPH(traffic$_{t_{curr}}, threshold$)
        $ED_{t_{curr}} \leftarrow CALC - ED(graph)$
        $CC_{t_{curr}} \leftarrow CALC - CC(graph)$
        $NND_{t_{curr}} \leftarrow CALC - NND(graph)$
        $Strength_{t_{curr}} \leftarrow CALC - Strength(graph)$
        $ED \leftarrow ED \cup ED_{t_{curr}}$
        $CC \leftarrow CC \cup CC_{t_{curr}}$
        $NND \leftarrow NND \cup NND_{t_{curr}}$
        $Strength \leftarrow Strength \cup Strength_{t_{curr}}$
        $t_{curr} \leftarrow t_{curr} + t_{sample}$
    end while
    return $ED, CC, NND, Strength$
end procedure
```

Table 3.1: Complexity indicators for G_1

	ED	Strength	CC	NND
AC1		0.2	0	4
AC2		1.1	0.2	2.63
AC3		1.1	0.33	3.27
AC4		1.2	0.33	3.33
AC5		1.4	0.33	3.14
Average	0.25	1.0	0.24	3.28

its vertices is 0. This is not the case for G_1, where there are several triplets, i.e. clusters, thus the average CC is not zero, specifically 0.24.

As it has been previously mentioned, NND measures how connected the neighbors of a vertex are. Let us consider AC_2. By removing an edge between AC_3 and AC_5 (which has a relatively big weight), we can see the difference in NND score. The NND impacts the measures taken to lower complexity. If AC_2 and AC_3 are moving closer to each other, the strength of their interdependency will increase. Moreover, given that there is an interdependency between AC_3 and AC_5, the situation will be more complex. Having this information, a change in trajectory would be proposed

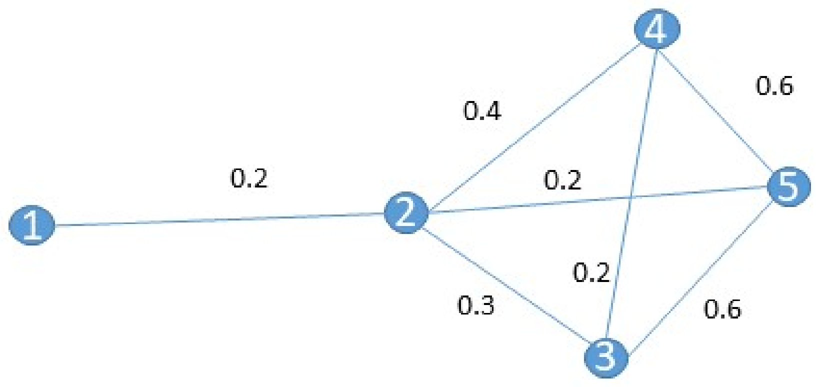


Figure 3.5: A complex graph G_1.

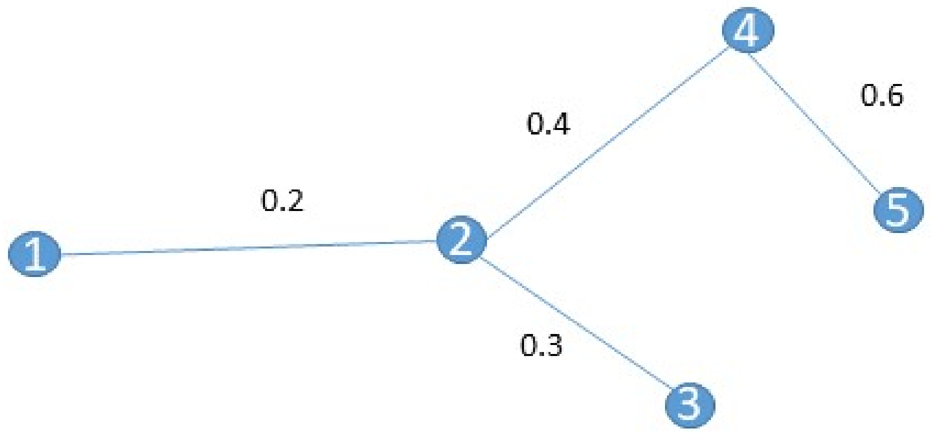


Figure 3.6: A simpler graph G_2.

in order to decrease complexity.

Figure 3.7 shows a more realistic example scenario. There, four aircraft are moving towards each other. In this example, they are all A320, flying at 30000 feet, with a speed of 300 kts. The distance between the furthest aircraft is around 240 NM, while the distance between the closer aircraft is around 120 NM. The threshold for interdependencies is set to 100 NM. The simulation is run for 20 minutes. The figure shows snapshots of the traffic in different time staps, at the beginning, at 8 minutes and at 16 minutes. At each point in time, the graph and the indicator scores are shown. While evaluating the merits of the proposed indicators is paramount,

Table 3.2: Complexity indicators for G_2

	ED	Strength	CC	NND
AC1		0.2	0	3
AC2		0.9	0	1.44
AC3		0.3	0	3
AC4		1.0	0	1.8
AC5		0.6	0	2
Average	0.15	0.6	0	2.25

an important aspect to investigate is the way the information is presented to the controllers. In this instance, we present a simple and intuitive way which presents the most crucial aspects of the graph. However, more research is needed to determine the best way such information must be presented.

Figure 3.8 shows the evolution of indicator in the simulation time. For the first 3 minutes, the scores are all zero, as the aircraft have not yet started creating interdependencies amongst them. The first interdependencies are created between $AC_1 - AC_3$ and $AC_2 - AC_4$. In this example, there was no intervention to separate the aircraft, therefore the peak for ED and Strength is reached around the 15 minute mark, where all aircraft are in conflict with each other, as demonstrated by the ED value of 1, which is the theoretical maximum. Interestingly, the peak values of CC and NND are reached around the 10 minute mark. Such a result indicates that all aircraft now have interdependencies with each other, shown by the value of NND being 3. This example illustrates how complex scenarios can occur even before the start of conflicts, as evidenced by the peak of CC and NND happening before the peak of ED and Strength.

Miles-In-Trail

A realistic example is the intersection of two Miles-in-Trail (MiT) flows. This way of traffic organisation is well accepted among controllers to reduce air traffic complexity. In MiT, traffic is structured into flows of aircraft following the same path. Flights in the same path are separated by a certain distance and their speed is regulated. These flows create queues of aircraft that are easier to manage by controllers [76]. In this section, we show different conflict scenarios and how they would be solved by different CR algorithms and we will illustrate how the values of each complexity indicator evolve during that time.

To do so, we adapt the scenario of Breil et al. [76]. They consider a flow that goes

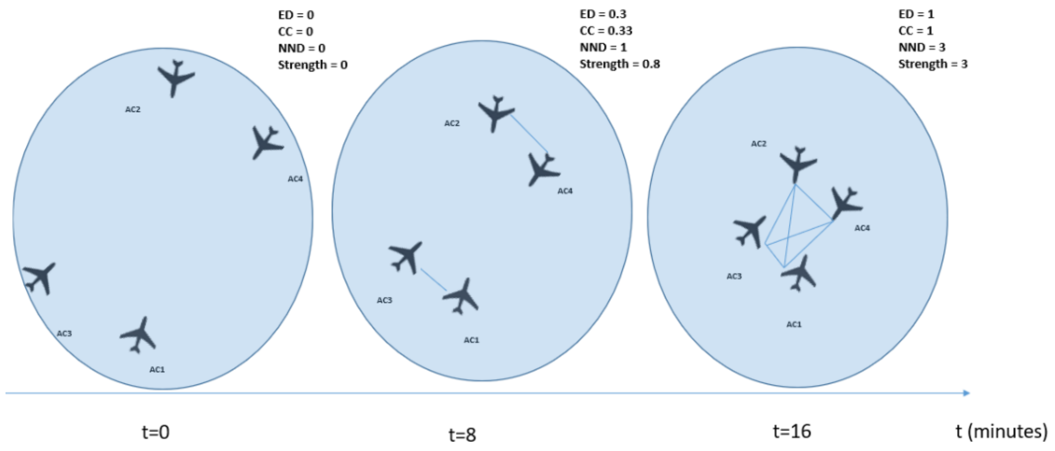


Figure 3.7: A more realistic example scenario.

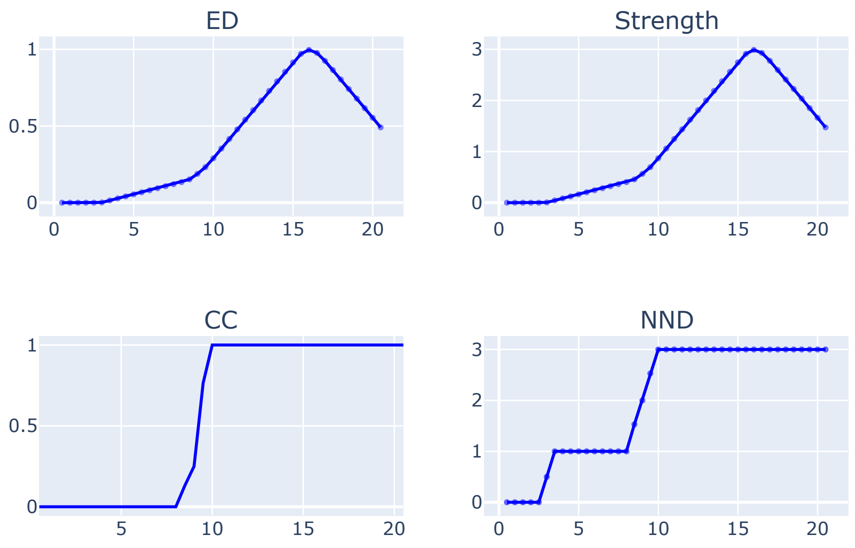


Figure 3.8: Indicator scores for the example scenario.

through waypoints LMG, MEN, MRM and another that goes through waypoints TOU, MEN and LYS. Aircraft are generated every 100 s from LMG and TOU at 20000 ft. The distance between aircraft in the same flow is kept at 14.14 NM. The interdependency threshold is set at 15 NM, so aircraft at the same flow always form interdependencies amongst them. Using these parameters, conflicts are induced in crossing aircraft from different flows.

In their work, Breil et al. [76] model air traffic as a multi-agent system and propose an algorithm to solve conflicts by local speed changes (henceforth referred to as MAS Speed). The goal of the algorithm is to choose an action from the action space to either solve a conflict, or which leads to the latest start time of a conflict. Furthermore, they extend it to include heading changes as well (henceforth referred to as MAS Heading). Differently from their work, we consider the speed changes separately from heading changes, and do not combine the two different ways of regulating a MiT network. For a more detailed description of the algorithm, we refer the reader to [76].

Furthermore, we adapt the Modified Voltage Potential (MVP) conflict resolution algorithm proposed in [77]. The MVP algorithm models conflicting aircraft as identically charged particles that repel each other in such a way that the conflict does not occur. The result is a displacement vector that is used to compute changes in the speed of conflicting aircraft. We refer the reader to [77] for a detailed description. Finally, we also show the case where no intervention is made, for reference.

Figure 3.9 shows the MiT network we consider. In the case of MAS Speed, aircraft must decide between three actions: cruise, accelerate, decelerate. Such a decision must be made every 5 seconds and the algorithm is validated greedily for each aircraft in parallel. Acceleration and deceleration are fixed to $\pm 4000 NM/m^2$. For the MVP algorithm, whenever there is a conflict pair, speed changes are applied to solve the conflict.

The scenario of MAS Heading is shown in Figure 3.10. In this scenario, aircraft can be put on parallel track to enable them to cross the intersection conflict free. Each flow is divided into three tracks separated by 5 NM. Aircraft must choose which track they must take. Finally, the sub-flows are merged into a single outgoing flow.

In this section, we consider 4 different scenarios: 2 aircraft per flow, 4 aircraft per flow, 8 aircraft per flow and 16 aircraft per flow. Four intervention methods are examined: MAS Speed, MAS Heading, MVP and a reference case (no intervention). Figure 3.11 shows the indicators for the case with 2 aircraft per flow. When an action is chosen to reduce the effect of the conflict, the CC and NND scores drop dramatically. This result indicates that our metric might be able to capture perceived controller complexity. Nevertheless, we see a huge spike for edge density and strength in the case of MAS Speed. This is attributed to the fact that the algorithm is

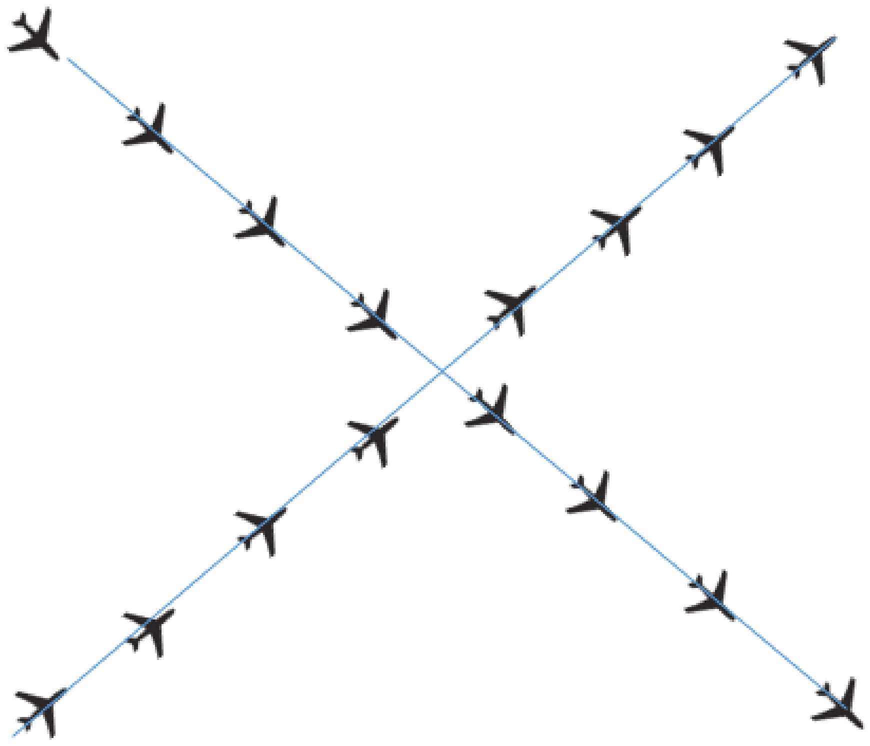


Figure 3.9: Miles-in-Trail scenario for the case of speed changes.

not able to solve this conflict. The MVP algorithm has a similar ED score to when there are no interventions, but the strength score behaves differently, which is due to the fact that the conflict is solved once it is detected. This is further confirmed from the comparison between MAS Speed and MVP.

Figure 3.12 shows the indicators for the case of 4 aircraft per flow. The results here are similar to the previous case. However, we note that in this scenario, MAS Speed does not lead to a spike in ED and strength. Nevertheless, MVP leads to a higher ED and strength score. MAS Heading leads to a more complex solution of the MiT scenario when 4 aircraft per flow are present, with only the NND not resulting in a spike.

Figures 3.13 and 3.14 provide similar information. In these two scenarios, where the number of aircraft per flow is quite high, the speeding changes algorithms lead to a noticeably lower scores for CC and NND. With many present aircraft, changing the heading might lead to a more chaotic situation. There is a bigger possibility of

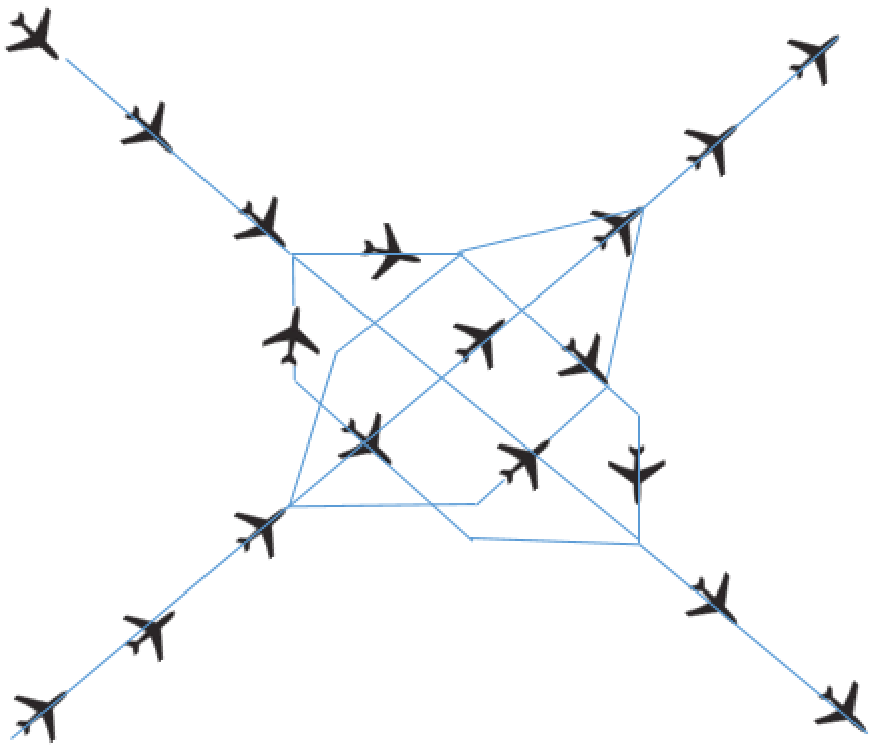


Figure 3.10: Miles-in-Trail with heading changes.

aircraft creating interdependencies with other aircraft that they would not have if the traffic was regulated by speed changes. Furthermore, we note that in all scenarios, the spike in the MAS Heading algorithm happens later then in the case of MAS Speed and MVP. This is an indicator that the bigger interpendencies happen when traffic is rerouted to its original flow. The indicators provide elaborate information to controllers, as they can be alerted that a complex situation will start. In situations with many present aircraft, heading changes are more difficult for a controller to handle, which is further evidence that our indicators are suitable to be used in real traffic. This analysis shows that the indicators can be used to assess how conflict resolution algorithms affect the complexity of the airspace, which is a dimension traditional CR algorithms do not consider.

Moreover, the proposed indicators can be used to decide appropriate solution techniques for given scenarios. For example, in cases with fewer aircraft, heading changes lead to lower complexity, while in scenarios with many aircraft, speeding changes might be preferred. Finally, the indicators can be used to design novel conflict resolution algorithms. Such algorithms can be used to not only solve conflicts,

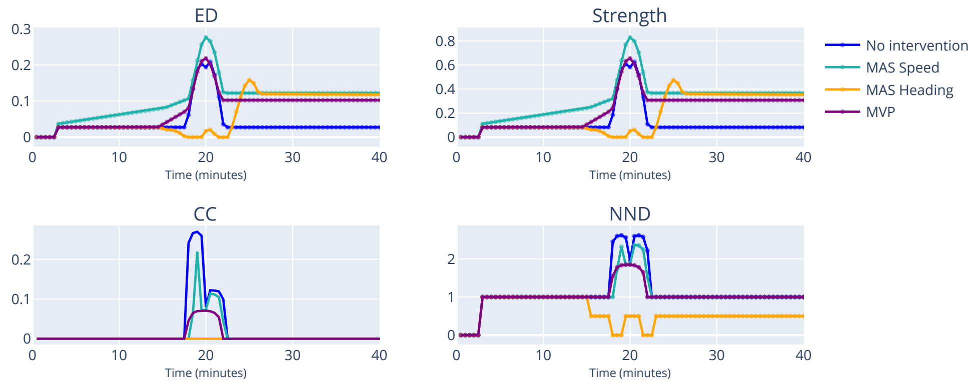


Figure 3.11: Miles-in-Trail with 2 aircraft per flow.

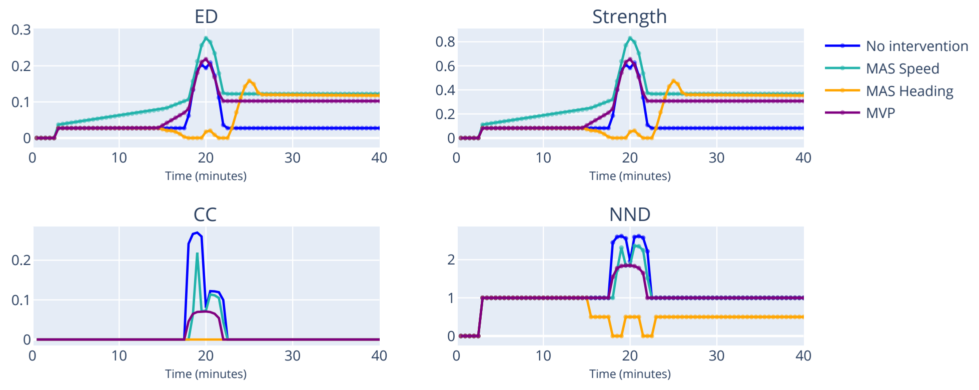


Figure 3.12: Miles-in-Trail with 4 aircraft per flow.

but also improve the quality of solutions. For instance, an algorithm based on the indicators could propose solutions that at best improve the complexity of the sector and at worse do not increase complexity, in addition to solving the present conflicts.

3.6 Evaluation on Real Traffic

Data

We evaluate these indicators on a day of flights taken from DDR2 (m3 file format consisting of actual trajectories). We take sector configurations of that day and eventually end up with 178 sectors, all in the European Civil Aviation Conference

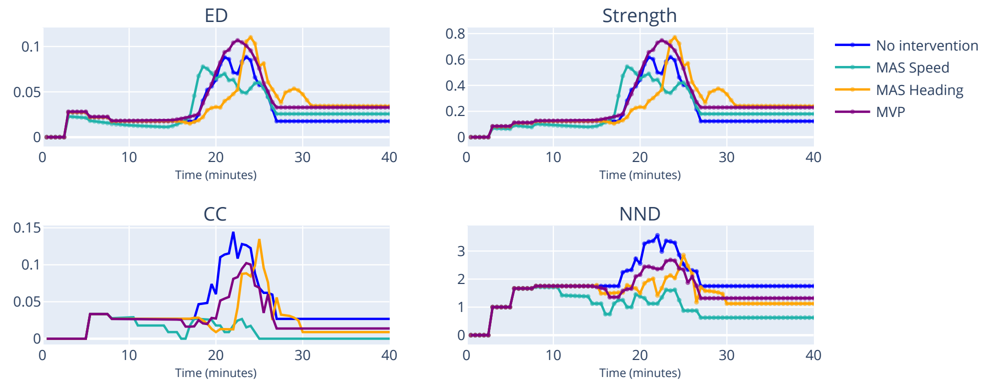


Figure 3.13: Miles-in-Trail with 8 aircraft per flow.

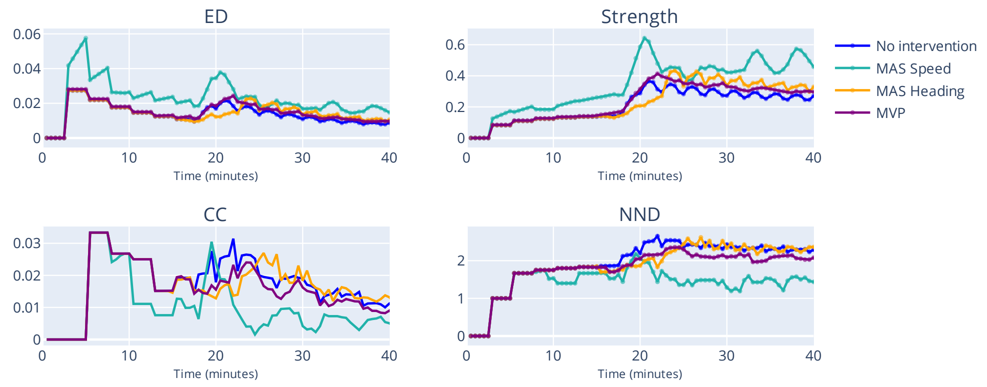


Figure 3.14: Miles-in-Trail with 16 aircraft per flow.

(ECAC)[1] area. Sector borders are kept unchanged throughout the day. The flights and sectors that are used in this work are en-route, i.e. active at 25000 feet and higher.

In order to determine the horizontal and vertical thresholds through which we define interdependencies, we do a system wide analysis of pairwise aircraft horizontal and vertical distances. As such, for each individual sector, we compute the average pairwise horizontal and distance based on data from the previous day. The thresholds

[1]`https://www.ecac-ceac.org/`

are then set as the mean pairwise distance (horizontal and vertical). In this way we adapt the thresholds for each sector, expecting that historical data can be useful to identify some traffic features if the operational context is similar. To those distances, we add as a buffer a value of 5 NM horizontally and 1000 feet vertically. Interdependencies should not be defined universally, as sector size plays a role in the distances or aircraft. Using such a method, we ensure that horizontal and vertical thresholds are set for each sector differently.

To get the complexity scores for the indicators, we sample the data every 30 seconds for the whole day.

Complexity Indicators

In this section, we visualize the complexity indicators for an hour of operations. The sector we chose had, on average, the most aircraft present at the same time. In Figure 3.15, we also visualize the occupancy in the the sector at each time step. This can be thought of as an extension of this metric into the time domain. Occupancy is one of the simplest yet most used classical complexity metrics. This section presents a comparison of this metric with the proposed indicators and we show that occupancy cannot capture nuances in aircraft interdependencies and thus complexity.

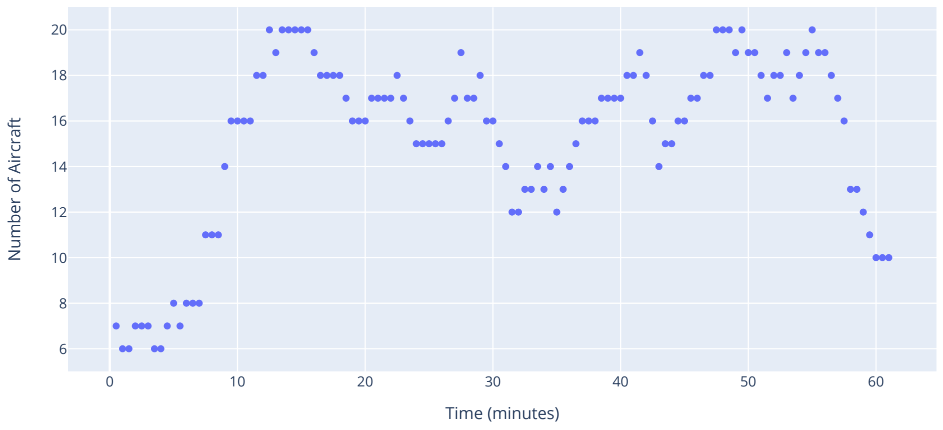


Figure 3.15: Number of flights present in the sector.

In Figure 3.16, ED of the hour of operations is shown. The ED score is relatively low, with the highest being around 0.6, which occurs when the number of aircraft is around 8. This means that the majority of aircraft do not get close enough to

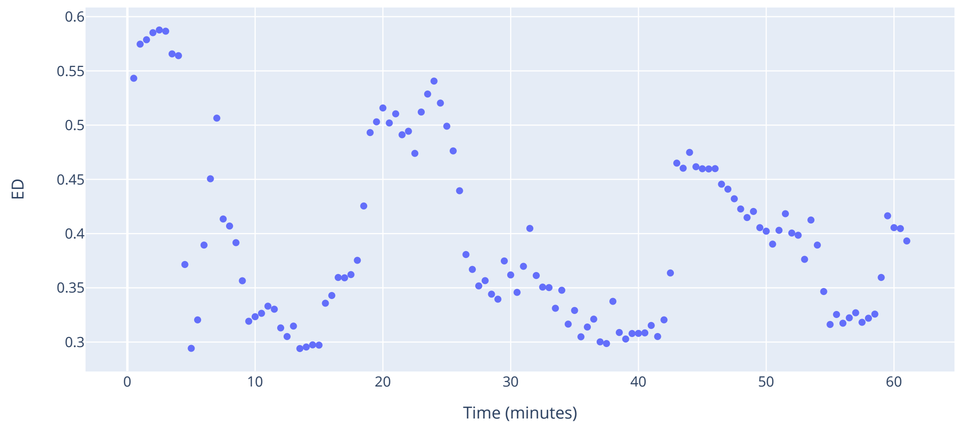


Figure 3.16: Edge density.

each other to be considered interdependent. Such a result further confirms initial statement, that the number of aircraft does not present a full picture of complexity.

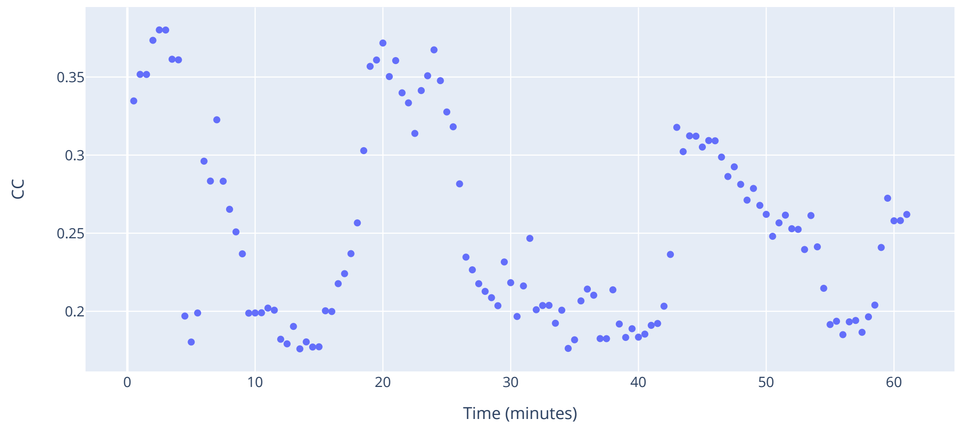


Figure 3.17: Clustering Coefficient.

Figure 3.17 shows the clustering coefficient throughout the hour. The overall distribution is similar to that of ED, which suggests that there is an area in the sector where aircraft tend to cluster. Nevertheless, one can see that such clusters do not last long (in this case around 5 minutes).

The NND score, shown in Figure 3.18, provides similar information. There is a peak around the 20 and 50 minute marks, which corresponds to a peak in the other

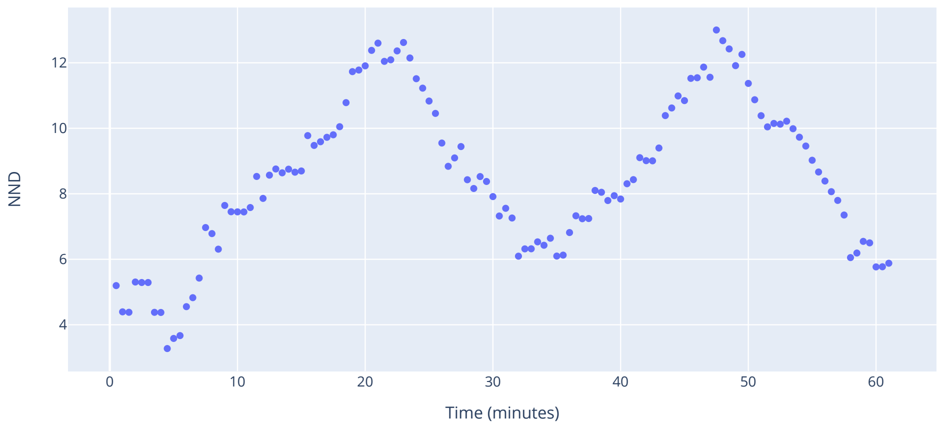


Figure 3.18: Nearest Neighbor Degree.

indicators as well as the number of aircraft. However, in the beginning, when the CC and ED score quite high, NND is at its lowest. This is due to a small graph size, which is further supported by a small number of aircraft. Between the two peaks, a steady decline and increase of NND is observed. Such a behavior 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 for the hour of operations is visualized in Figure 3.19. Similarly 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 in the previous paragraph about the nature of interdependencies. On top of that, the similarity between NND and strength further suggests that in this particular case, not only do interdepedencies span multiple aircraft, they are also quite strong, which adds to the complexity of the situation.

Correlation between number of aircraft and the indicators

In this section, we show that the combination of the proposed indicators can capture more detailed information than the number of aircraft, which is the simplest complexity indicator. To do this, we calculate the Pearson correlation coefficient for all sectors. Table 3.3 shows the mean correlation for each indicator.

We notice that the Pearson correlation coefficient for ED is 0.39 value which means

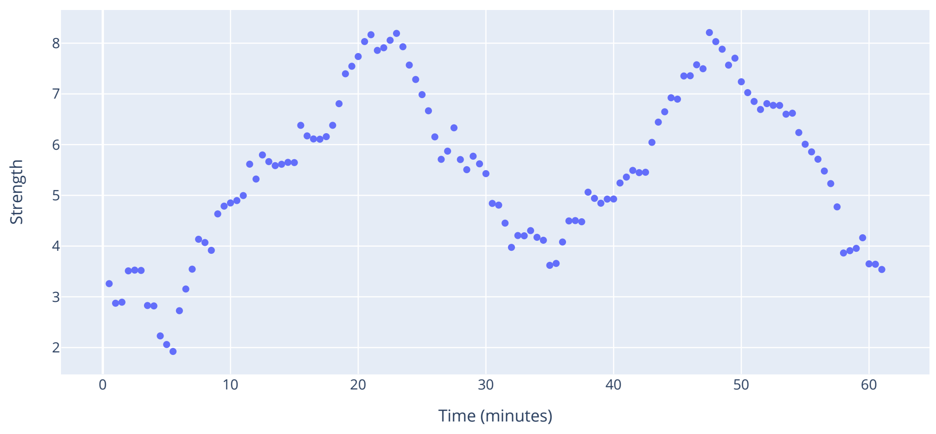


Figure 3.19: Strength.

Table 3.3: Mean correlation of each indicator with the number of aircraft

ED	Strength	CC	NND
0.39	0.86	0.64	0.88

Table 3.4: Mean correlation between indicators

	ED	Strength	CC	NND
ED		0.65	0.54	0.61
Strength	0.65		0.83	0.98
CC	0.54	0.65		0.82
NND	0.61	0.98	0.82	

there is low correlation. This result is further evidence that number of aircraft does not account well for sector complexity.

The correlation of aircraft count and CC and NND is higher. Especially, in the case of NND the correlation coefficient is more than 0.88, indicating high correlation. This behavior is consistent with results from the previous section, which means that when there are more aircraft in the sector, interdependencies tend to span multiple aircraft, while being less likely to form clusters.

A relevant result is that CC correlates to the number of aircraft more than ED. When a sector is more densely populated, clusters tend to form even when only a fraction of present aircraft are interdependent. This shows that even when a high number of aircraft correlates with more complex situations, our indicators give more information about the nature of this complexity.

As previously stated, the evolution of strength through time demonstrates that when there is a high number of aircraft present in the sector, interdependencies tend to be stronger. The results shown in Figure 3.19 verify this claim.

Correlation between indicators

In order to show that all indicators are needed, we calculate how they correlate with each other and interpret these results, shown in Table 3.4

We notice that correlation between CC and ED is low, an expected result from the results of the previous section. This is additional evidence that even small graphs can be quite complex. The mean correlation between ED and NND serves are further confirmation.

The correlation between NND and CC is high. Both of these indicators are defined to give higher scores to more interconnected graphs, thus this is an expected result. Nevertheless, these indicators inherently account for different topologies, as evidenced by Figure 3.18 and Figure 3.17.

The results indicate low correlation between the ED and strength indicators. As we have noted before, graph size (which ED measures), shows that not all aircraft present in the sector contribute to the overall complexity. Strength adds a layer of information by measuring the severity of interdependencies. The strength-ED correlation implies the presence of bigger graphs with loosely interdependent aircraft and smaller graphs with tightly interdependent aircraft.

The strength-NND and strength-CC correlations are quite high. This suggests that in the presence of tighter interdependencies, clusters and other forms of multi-cluster interdependencies are very likely to form. However, as shown in the MiT example, very different geometries with the same strength can emerge. Therefore, to capture such topological information and combine it with the severity of interdependencies, it is important to consider all indicators.

3.7 Conclusions and Future Work

In this paper, we formalize four indicators based on graph theory to measure sector complexity. 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 indicate that the proposed indicators give detailed information on complexity and overcome drawbacks of existing metrics.

We evaluate the indicators in synthetic traffic geometries, as well as in a standard situation such as Miles-in-Trail. For the latter, we adapt several conflict resolution algorithms and show how different resolution strategies can affect the overall complexity. We argue that complexity is a factor conflict resolution algorithms should consider and through results show different strategies might be preferred depending on the number of aircraft.

Furthermore, the proposed indicators are assessed using a day of flights taken from DDR2 data. Sector configuration of that day is taken and we monitor 178 sectors in the ECAC area. Results show that the indicators are more effective in capturing geometrical complexity than aircraft density. The information obtained from using the indicators can provide the controllers with a better understanding of complex areas inside the sector. On top of that, we demonstrate that the four indicators express different facets of complexity, confirming that all indicators are needed.

The indicators are evaluated using real traffic by taking a one hour window with samples every 30 seconds which results in a stable output. Nevertheless, the length of the time window must be investigated further. Several factors must be considered when deciding on the length of the window. One of the most important factors is including uncertainty in trajectory prediction, which we do not consider in this work. As such, suitable time windows might be smaller, or might require different sampling strategies.

As previously stated, the indicators provide a new framework in the design of conflict resolution algorithms in order to preserve safety while reducing traffic complexity. Complexity is usually not taken into account in designing such algorithms, which can lead to solutions that can make the situation difficult for the controller to handle. Additionally, by using the indicators, algorithms can be tuned to encourage resolutions that lead to lower complexity and discourage those that increase complexity. Such algorithms can be used to improve the quality of resolutions, in addition to solving the present conflicts. For instance, Isufaj et al. [91] propose a multi-agent reinforcement learning approach to conflict resolution which considers airspace complexity as one of the factors that the model must optimize in addition to solving conflicts. The indicators proposed in this work could allow for a more granular optimization of complexity by providing more detailed information.

In addition, by modeling traffic as a graph, we open the door to further applications of graph theory in aviation. First of all, this work should be a baseline for designing complexity indicators based on graph theory. As such, future work should consider different methods for the definition of interdependencies and the value of thresholds. For instance, machine learning methods can be used to generate the graph [92]. Furthermore, U-Space [93, 94] is envisioned to be a set of services to provide ATM to unmanned aviation in Europe. One of its key services will be dynamic capacity management [93], used to for strategic conflict resolution. The indicators proposed in this work, can be adapted to this use-case and serve as an implementation of this service.

Finally, a very important continuation would be the mapping between the indicators and controller workload, which is an ongoing topic in the ATM community. Ways of measuring the workload could be through subjective scores or more sophisticated methods such as EEG [38, 39]. Such a work would 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 this 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 various factors.

Chapter 4

FROM SINGLE AIRCRAFT TO COMMUNITIES: A NEUTRAL INTERPRETATION OF AIR TRAFFIC COMPLEXITY DYNAMICS

Present air traffic complexity metrics are defined considering the performance indicators of different management layers of ATM. These layers have different objectives which in practice compete to maximize their own goals, which leads to fragmented decision making. This fragmentation together with competing KPAs requires transparent and neutral air traffic information to pave the way for an explainable set of actions. In this paper, we introduce the concept of single aircraft complexity, to determine the contribution of each aircraft to the overall complexity of air traffic. Furthermore, we describe a methodology extending this concept to define complex communities, which are groups of interdependent aircraft that contribute the majority of the complexity in a certain airspace. In order to showcase the methodology, a tool that visualizes different outputs of the algorithm is developed. Through use-cases based on synthetic and real historical traffic, we first show that the algorithm can serve to formalize controller decisions as well as guide controllers to better decisions. Further, we investigate how the provided information can be used to increase transparency of the decision makers towards different airspace users, which serves also to increase fairness and equity. Lastly, a sensitivity analysis is conducted in order to systematically analyse how each input affects the methodology.

4.1 Introduction

Air Traffic Management (ATM) is a complex socio-technical system comprised by three main layers; air space management (ASM), air traffic flow management (ATFM) and air traffic control (ATC) [95], whose performance is measured through various Key Performance Areas (KPAs), of which some of the most important are safety, capacity, cost-efficiency and environment [5]. Although part of ATM structure, each of these layers have different objectives which in practice compete to maximize their own goals. Several authors point out that complex interdependencies among the decision layers can cause unnecessary penalization on some KPAs to improve others. In [96] authors claim as a result of early SESAR projects such as STREAM [6] that the exact relationship among these KPAs is still not well understood and should be further studied in future research. Moreover, recent finalized projects such as APACHE [7] targeting the analysis of the interdependencies between the different KPAs by capturing the Pareto-front of ATM, still claims that further research is needed to uncover the inter-dependencies between the different KPAs [97]. An important challenge that must be overcome to reach such a Pareto-front of ATM KPAs is the lack of an effective coordination mechanism (i.e., system behaviour) among ATM subsystems and corresponding Decision Support Systems (DSS).

Figure 1 illustrates a conceptual framework for ATM-related quality of services based in [98, 99]. The idea is to visualize all the components (i.e., subsystem, KPA, DSS) and mark the inter-relations between them. The distinction of components is done by color-coding based on their functions and definitions. Each component is connected to one or other components. The component that is attached to the arrow-head indicates that it is the one which is affected by the other. Note that connectors have different colours, based on the origin of the subsystems (e.g., ATC). According to [100], the represented interactions can be considered as complex adaptive systems (CAS), which typically include multiple loops and multiple feedback paths between many interacting entities, as well as inhibitory connections and preferential reactions.

Furthermore, an illustrative example highlights the interactions among the DSS that target the airspace management (AM) and the trajectory management (TM) and capacity management. This is a common example to highlight how these decisions can cause a chain of reactions resulting in en-route ATC inefficiency, delays at airport (taking-off and landing) and en-route ATFM inefficiency [101]. To date, the lack of a formal analysis between these control mechanisms leads to a lack of transparency and coordination between different DSS that could improve ATM performance.

Complex interdependencies among the mentioned KPAs have been key for the aeronautic community to accept the confusing term *emergent dynamics* which in fact is justified by the un-modelled behavioural dynamics among these objectives (i.e., capacity, safety, and efficiency). For instance, consequences of a small reduction of a sector capacity (considering ground weather conditions) usually is tackled by over-conservative ATM actions to avoid safety issues at a cost of penalizing flight

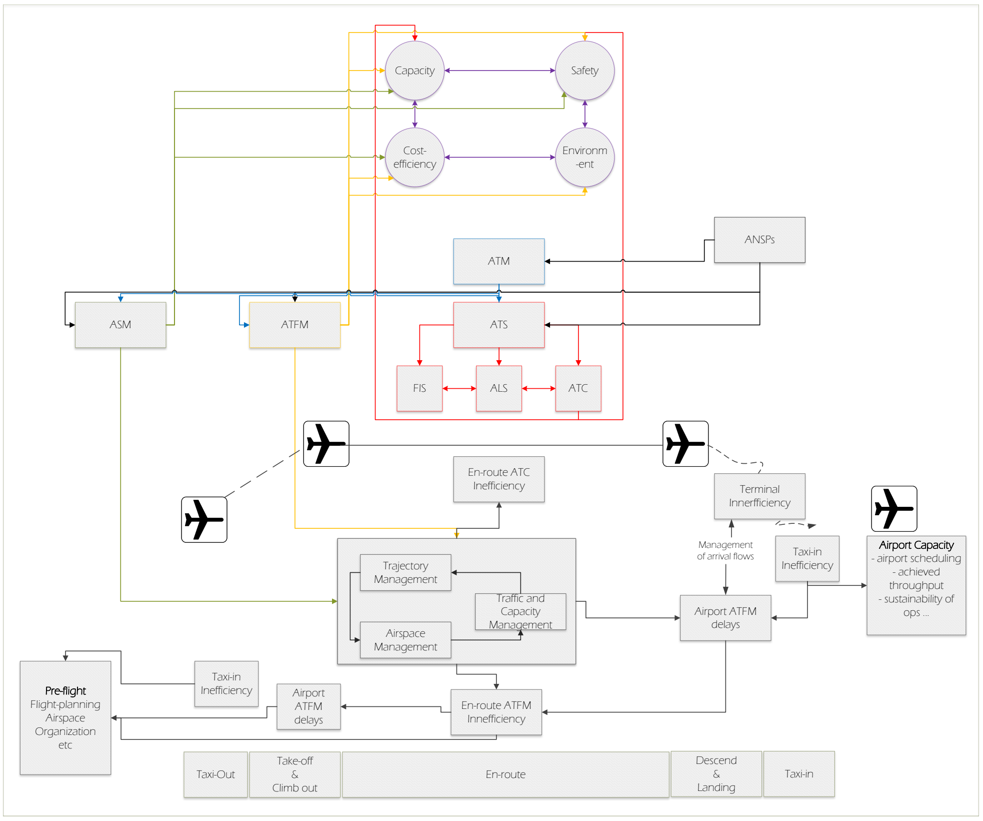


Figure 4.1: Conceptual Framework for ATM-related quality of services.

efficiency. In the other hand, other solutions with better information (weather information at flight deck) could improve flight efficiency. In [102] a difference between *weak emergence dynamics* and *strong emerge dynamics* is introduced to differentiate between micro-level interactions among subsystems (weak) and emergence caused by irreducible macro causal mechanisms (strong). Authors of this paper, accept that unpredictable decision-making processes carried by a human actor (i.e. aircraft pilot, air traffic controller) justify the term of *strong emergent dynamics* in ATM which can be observed in several ATM socio-technical subsystems [103].

To avoid an ATM system dynamics ruled by *strong emergent dynamics* due to abrupt human behaviour, in this paper a new methodological framework to enhance a common understanding among the different stakeholders is proposed. Worthwhile to highlight that as a result of the implemented methodology, the framework allows ANSP work closely with the rest of stakeholders avoiding over constraining solutions. Moreover, the proposed framework paves the way for a shared situational awareness in which the effects of unpredictable decision-making processes carried by human actors is mitigated by the consensus reached among the different actors, transforming the *strong emergent dynamics* into *weak emergent dynamics* of ATM.

The core idea behind the proposed methodology, lies in understanding complexity evolution of the overall air traffic. In this paper we extend the complexity notion and indicators proposed in [104], by introducing single aircraft complexity concept. We define single aircraft complexity as the contribution that individual aircraft have to the overall sector complexity. Furthermore, we identify groups of interdependent aircraft that form high complex spatio-temporal areas. This method identifies the individual and community (i.e., groups of aircraft) contributions in an on-line fashion and gives information about the creation, evolution and disappearance of communities. This information could enhance equity, fairness at aircraft or airline granularity level together with NM and ANSP service performance. The proposed method could make an immediate impact in the smooth transition between ATM layers and DSS tools. In other words, each subsystem should comply with their operational performance specifications (e.g., ATC must prevent loss of separations), while decreasing mutual penalization.

The rest of this work is structured as follows: In Section 4.2, we describe in detail the methodology. Section 4.3 provides an overview of the experimental setup. In Section 4.4, we present and discuss several use cases based on synthetic and real traffic, as well as an extensive sensitivity analysis. We draw conclusions and discuss future steps in Section 4.5.

4.2 Methodology

Spatiotemporal Graph-based Complexity Indicators

While there have been many different definitions for airspace complexity, in this work we will extend the one introduced in [104]. There, the authors focus on defining complexity for a certain volume of space (e.g., a sector) during a time window of interest and model air traffic as a dynamic graph $G(t) = (V(t), E(t))$. The set of vertices $V(t)$ for time t is comprised of the aircraft present in the sector at the time, while the set of edges $E(t)$ are the interdependencies between each pair aircraft at time t. Interdependencies are defined based on the distance between aircraft, more specifically, if two aircraft are closer than a certain threshold then there will be a weight between these two aircraft. The closer the aircraft are, the bigger will be the weight of the edge between these two aircraft, which means that the graph is **weighted** and **undirected**. Weights are normalized to be between 0 and 1.

In their work, they are interested for en-route traffic at the tactical level, therefore they define the weight of the edge to be maximal (i.e., 1) when there is a loss of separation between a pair of aircraft (5 NM horizontally and 1000 feet vertically). Horizontal and vertical interdependencies are calculated separately and the overall interdependency between two aircraft is the average of the two. Formally, this is defined as:

$$wh_{i,j}(t) = \begin{cases} 1 \text{ if } dh_{i,j}(t) \leq H \\ 0 \text{ if } dh_{i,j}(t) \geq thresh_h \\ \frac{thresh_h - dh_{i,j}(t)}{thresh_h - min_h} \text{ otherwise} \end{cases} \tag{4.1}$$

$$wv_{i,j}(t) = \begin{cases} 1 \text{ if } dv_{i,j}(t) \leq V \\ 0 \text{ if } dv_{i,j}(t) \geq thresh_v \\ \frac{thresh_v - dv_{i,j}(t)}{thresh_v - min_v} \text{ otherwise} \end{cases} \tag{4.2}$$

$$w_{i,j}(t) = \begin{cases} \frac{wh_{i,j}(t) + wv_{i,j}(t)}{2} \text{ if } wh_{i,j}(t) > 0 \;\;\&\;\; wv_{i,j}(t) > 0 \\ 0 \text{ otherwise} \end{cases} \tag{4.3}$$

where $wh_{i,j}(t)$ and $wv_{i,j}(t)$ are the horizontal and vertical weights at time t. Furthermore, $dh_{i,j}(t)$ and $dv_{i,j}(t)$ are the distances, H and V are the safety distances and $thresh_h$ and $thresh_v$ are the thresholds.

Airspace complexity is treated as a multifaceted notion and the authors propose four indicators that quantify topological information and combine it with the severity of the interdependencies. We will briefly describe these indicators, however we refer the reader to [104] for a detailed overview.

Edge Density (ED) measure how many edge the graph has compared to the number of edges in a fully connected graph of the same size with maximal edge. Formally:

$$ED(G,t) = \frac{\sum_{(i,j)\in E} w_{i,j}(t)}{A(V_t)}, A(V_t) = \frac{|V_t|(|V_t| - 1)}{2} \tag{4.4}$$

where $|V_t|$ is the number of vertices in the graph at time t and $A(V_t)$ is the maximal number of edges.

Strength measures the severity of pairwise interdependencies. It is obtained by extending the definition of vertex degree to account for edge weights:

$$s(i,t) = \sum_{j=1}^{N} w_{i,j}(t) \tag{4.5}$$

Clustering Coefficient (CC) measures the local cohesiveness and gives information regarding the local neighbourhood of each vertex (i.e., aircraft). Formally, it is calculated as follows:

$$CC(i,t) = \frac{\sum_{j,k}(w_{i,j}(t) + w_{j,k}(t))}{2 \cdot (s(i,t)(deg(i,t) - 1)}, \forall (i,j,k) \in \mathcal{T}(t) \tag{4.6}$$

Nearest Neighbor Degree (NND) calculates a local weighted average degree of the nearest neighbour for each aircraft:

$$NND(i,t) = \frac{\sum_{j=1}^{N} w_{i,j}(t) deg(j,t)}{s(i,t)} \tag{4.7}$$

The first indicator is inherently a global measure, while the remaining three indicators are turned into global measures by taking the average across vertices in the graph.

The overall complexity of the sector is chosen to be given as the evolution in time of each indicator and the authors argue that this results in a more nuanced overview of complexity.

Single Aircraft Complexity

While the previously described methodology gives a more nuanced view of complexity than simpler metrics (e.g., dynamic density [30]), it still suffers from a common drawback of the majority of existing complexity metrics: a lack of interpretability of the complexity scores. More specifically, given a certain traffic configuration, existing methods cannot provide information as to which areas of the sector are causing most of the complexity and how much of it they are causing. Furthermore, [104] do not discuss ways how to combine the information provided by each indicator.

In this work, we exploit an inherent characteristic of complexity defined based on graph theory to overcome this major drawback. As previously mentioned, three of the four indicators described, which will be in the focus of our work, can be defined in terms of single aircraft, with the overall being the average across aircraft. This means that we can generate a complexity score for every single aircraft present in the sector in time t and for every indicator without loss of information. However, this is not sufficient, as single scores would simply induce an order between aircraft for every indicator without providing any information regarding the overall situation of complexity in the sector. Another issue with this method is that it contains redundant information, as interdependencies are undirected, e.g.,

$A \rightarrow B$ and $B \rightarrow A$. Finally, it is not clear how to interpret the scale and value of the complexity scores. Furthermore, since the indicators have a different range of possible values that they can take, it is not trivial how to combine the individual absolute scores of every aircraft.

In this paper, we propose to slightly change perspective and calculate the contribution of each aircraft to the overall sector complexity. This results not in an absolute score, but in a percentage that is relative to what is currently happening in the sector at the time. Without loss of generality, we will show how the contribution is calculated for the *strength* indicator. Let us consider an arbitrary sector which at time *t* is occupied by the aircraft shown in Figure 4.2. Following Equation 5, we can determine the individual strengths of the aircraft, shown in Table 4.1.

The individual contributions are calculated as which part of the whole strength each aircraft is responsible for. Differently from the original definition of the overall strength (the average), in this work we slightly modify this definition and take the whole as the sum of all aircraft. In this case, the overall strength would be:

$$s(t) = \sum_{i=1}^{N} s(i,t) \tag{4.8}$$

The contribution of aircraft *i* to the overall strength would then be:

$$c_s(i,t) = \frac{s(i,t)}{s(t)} \tag{4.9}$$

Using this formula, we can now determine the contribution of each aircraft to the overall strength as shown in Table 4.1. Following a similar method we can generalize how to calculate the contribution of an aircraft for every complexity indicator relevant to our work (strength, CC, NND):

$$c_{\mathcal{I}}(i,t) = \frac{\mathcal{I}(i,t)}{\mathcal{I}(t)} \forall \mathcal{I} \in [strength, CC, NND] \tag{4.10}$$

This method allows us to meaningfully combine the contributions for all three complexity indicators. As we are considering the contributions relative to the current situation in the sector, we can observe what percentage of the overall sector complexity each aircraft is responsible for. While there are different ways this can be achieved depending on the use case, in this work we choose the average of non-zero valued indicators at the sector. Only the indicators that have a non-zero value are

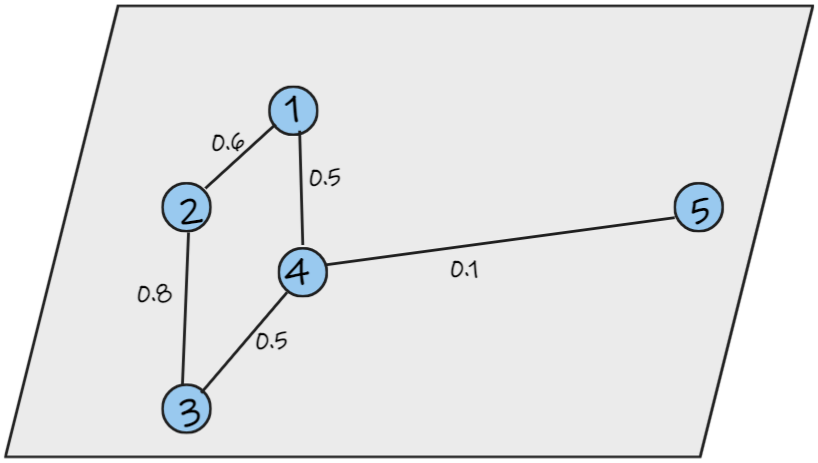


Figure 4.2: Example of an arbitrary sector.

Indicator	Value	Percentage
s_1	1.1	22%
s_2	1.4	28%
s_3	1.3	26%
s_4	1.1	22%
s_5	0.1	2%

Table 4.1: Individual contributions for the strength indicator.

used, as we are interested in showing the contribution to the existing complexity in the sector as quantified by the indicators. For instance, if $CC(t) = 0$ then this indicator is not a source of complexity for the sector at time t, therefore aircraft should not be attributed with contributing to this indicator. This is formalised as:

$$c(i,t) = \frac{\sum_{\mathcal{I} \in [strength, CC, NND]} c_{\mathcal{I}}(i,t)}{|\{\mathcal{I} \in [strength, CC, NND] : \mathcal{I} > 0\}|} \times 100 \qquad (4.11)$$

where the denominator is the cardinality of the set of non-zero valued indicators at time t.

This methodology is illustrated with the example in Figure 4.3, and the results in Table 4.2. As it can be seen, aircraft 2 contributes the most to the overall complexity with 20.45%. This example shows that the methodology successfully combines the information of all complexity indicators, as aircraft 2 is part of a cluster with 1 and 4 and also has a strong interdependency with aircraft 4 which increases its

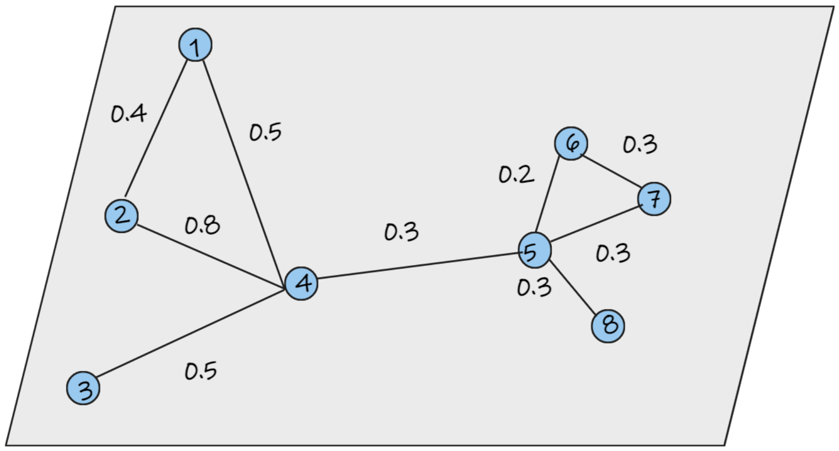


Figure 4.3: More complex example.

strength and NND scores. The topology of the subgraph comprised of aircraft 1-4 is mirrored from the subgraph formed of aircraft 5-8. However, we note that the interdependencies in the latter are weaker, which is correctly reflected in the contributions of these aircraft.

Nevertheless, we note again that the method of combining the contributions from each complexity indicator can depend on the use-case, using a weighted average instead. For instance, let us assume that this methodology will be used by the ATC. In that case, it could be reasonable that the most important indicator is strength, as it directly informs about the distance of the aircraft and how close they are to a loss of separation. Such information could be provided by weighing more the contributions from the strength indicator, while maintaining some of the information provided by the other two indicators.

Detection of Complex Spatiotemporal Communities

Using the concept of single aircraft complexity introduced in the previous section, it is possible to find groups of interdependent aircraft that contribute the majority of the complexity in the sector. Finding tightly interdependent groups of nodes in a graph (i.e., communities) is a well known problem in graph theory, with many existing algorithms providing high quality communities efficiently, such as the Louvain and Leiden algorithms [105, 106]. However, these algorithms optimize for *modularity*, which is a quantity that measures the density of connections within

Aircraft	Contribution
1	18.76%
2	20.45%
3	11.61%
4	14.23%
5	9.01%
6	11.12%
7	11.85%
8	10.22%

Table 4.2: Contribution to complexity from each aircraft in Figure 4.3

a community. Graphs with a high modularity score will have many connections within a community but only few pointing outwards to other communities. Briefly, the algorithms explore for every node if its modularity score might increase if it changes its community to one of its neighboring nodes.

Given the definition of modularity, it could be reasonable to expect that the communities that these algorithms find could coincide to high complexity communities, but the verification of this assumption will simply increase the runtime of the algorithm. Nevertheless, the more important issue of the aforementioned algorithms is that communities tend to share at least some edges between them. For the application in this paper, it is unclear how this situation ought to be treated.

To illustrate this point, let us take the graph in Figure 4.3. In this case, the Louvain and Leiden algorithms output two communities: one comprised of aircraft $\{1,2,3,4\}$ and the other comprised of aircraft $\{5,6,7,8\}$. However, these communities are connected, as there is an edge between 4 and 5. It is non-trivial to determine what weight is big enough to consider both communities together. Therefore, in this work we choose to be more conservative, considering always such communities together. In fact, this is also a known problem in graph theory known as *connected component* detection [107]. In graph theory, a connected component of a graph is a connected subgraph that is not part of any larger connected subgraph. Such components separate the vertices into disjointed sets. In the case of Figure 4.3, there is one connected component, i.e., the whole graph. An advantage of this choice is that connected components can be determined in linear time $O(n)$ [107] where n is the number of vertices, while the Louvain algorithm runs in $O(m)$ where m is the number of edges, which is typically considered slower as the number of edges is higher than the number of vertices [108]. In this work, we filter out communities with one aircraft.

Following Equation 11, we can determine the contribution of the community to the overall sector complexity, which we define as the sum of contributions for the individual aircraft in the community. Determining when a community is in fact a complex community is not trivial. Complexity has often been linked to the workload of controllers [73], which can be subjective [37, 38, 39]. In this work, we do not attempt to make any claims to relate the complexity indicators with the workload of controllers, we merely present a methodology that provides granular information of complexity given the definition of the aforementioned complexity indicators (which are objective). Therefore, in this work, we propose setting a contribution threshold above which a community is deemed to be a complex community. In such a way we maintain some flexibility in defining complex communities to better fit decision makers at the present structure of ATM, allowing each user to set a particular threshold. Formally, for a community C in time step t, we have:

$$complex(C, t) = \begin{cases} True & \text{if } \sum_{i \in C} c(i, t) \geq thresh \\ False & \text{else} \end{cases} \tag{4.12}$$

where $thresh$ is the user/problem specific threshold.

So far, we have defined complex communities only for one time step t. However, as we are looking at a time window (here we assume that the time window is a series of discrete time steps), the continuous evolution of traffic complexity should be analyzed. We have determined three generic events that could happen: appearance, disappearance and evolution.

Appearance and disappearance of a complex community are defined as the time steps in which the community started being and stopped being a complex community. These two events can be trivially determined by applying Equation 12 for the length of the time window. On the other hand, the evolution of complex communities requires more consideration. During the period of time when a complex community exists, new aircraft might join it, i.e., at least one existing aircraft of the community forms an interdependency with an aircraft outside of it; or leave it, i.e., an existing aircraft stops having any interdependencies with the other aircraft of the community. In such cases, the community should be considered the same, which in this work we will refer to as having the same *label*. Therefore, the problem of community evolution can be seen as determining how community labels are maintained in time.

In order to formalize the evolution of complex communities we propose an algorithm based on the Jaccard similarity. This method, formally defined below for two arbitrary sets A and B:

$$\mathcal{J}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.13}$$

measures similarity of sets as the size of intersection between the size of the union of these sets. The range of $\mathcal{J}$ is between 0 and 1, with 0 meaning that the intersection is empty, i.e., the sets have no common elements. In this work, we utilize Jaccard similarity to determine the evolution of complex communities through time. For an arbitrary community C in time t, we determine if there are communities in $t - 1$ that are similar to it. If in fact there are multiple communities that have a non-zero similarity to C, then the one with the biggest similarity score is determined to have the same label as community C. In more concise terms, if communities share some members in consecutive time steps, they are defined to share the same label feature of this algorithm is that it is only necessary to look in the previous time step, as labels can be propagated through time. Furthermore, it is trivial to determine at what time step aircraft joined or left an existing community.

If there are no similar communities in $t - 1$ then community C is a new label in the set of complex communities for time window we are studying. Consequently, we can define all three generic events for complex communities in terms of labels: appearance is the first time when a label is present and disappearance is the last time when the label is present. The whole algorithm is described in Algorithm 2.

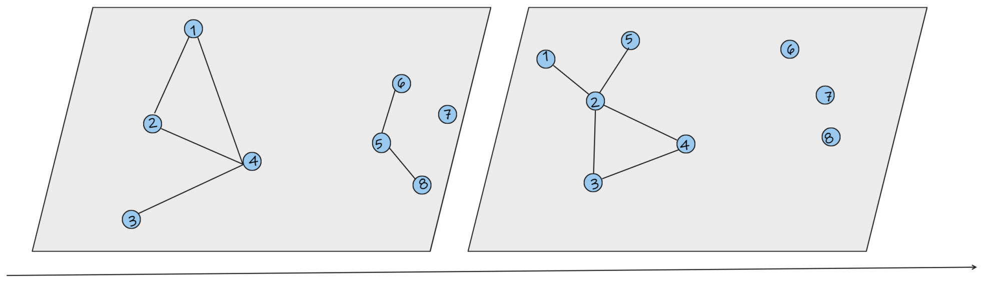


Figure 4.4: Illustration of complex community detection algorithm.

Let us illustrate this algorithm through the example in Figure 4.4. There, an arbitrary sector in two time steps is shown. In t_1, there are three communities, namely the community labelled C_1 with aircraft 1-4, then community C_2 with 5-7 and the last one is community C_3 with a single aircraft, 8, which is filtered out. Let us assume

Algorithm 2 Detection of Complex Communities

```
  $t \leftarrow [1, ..., T]$                                ▷ Time window that we are studying

  $Comm_{complex} \leftarrow \{\}$                          ▷ Set of complex communities, initialized as empty
  while $t \leq T$ do
Require: $G_t$                                              ▷ Assume we have the graph induced by the traffic in t
      $Comm_t \leftarrow communities(G_t)$                  ▷ Find all communities for the current time step
      For Each $C \in Comm_t$ do
          if $complex(C, t)$ then                           ▷ Equation 12
Require: $Comm_{complex}[t-1]$                              ▷ Get complex communities in t-1
              For Each $C_{t-1} \in Comm_{complex}[t-1]$ do
                  $similar \leftarrow \mathcal{J}(C, C_{t-1})$
              end for
              $similar_{max} \leftarrow argmax(similar)$    ▷ Find most similar community
              if thenNo $similar_{max}$                     ▷ No similar communities found
                  Add C to $Comm_{complex}$                 ▷ Add new label to complex communities
              else
                  Update $similar_{max}$                    ▷ Most similar community gets data from C with
                                                            ▷ added, removed members and time step when this happened
                  Update $Comm_{complex}$
              end if
          end if
      end for
  end while
  return $Comm_{complex}$
```

that in t_1 community C_1 is complex while the other are not. As we are at the initial time step, there are no previous time steps, thus the set of complex communities would have C_1 with members 1-4 all added at t_1.

In t_2, the communities have changed in terms of membership. Community C_4 with members 1-5 is a complex community. As per the algorithm, the previous time step would be queried to find any other existing complex communities, where C_1 would be found. Then, the Jaccard similarity would be calculated with $\mathcal{J}(C_1, C_4) = 0.8$. Therefore, there exists one complex community in the previous time step that has a non-zero similarity score with C_4. This means that C_4 received C_1 as the label and the original community is updated to contain the new information. Thus, C_1 now contains aircraft 1,2,3,4,5 with the former 4 aircraft being added in t_1 and aircraft 5 being added in t_2. For the sake of completeness, let us investigate what happens to community C_2 in t_2. As it can be observed, the remaining aircraft that comprised C_2 have no interdependencies with any other aircraft. Therefore, there are no communities with non-zero similarity to C_2. In this case, if C_2 was indeed a complex community we would be able to say that it appeared in t_1 with members 5,6,7 and disappeared in t_2.

4.3 Experimental Setup

In order to effectively showcase the algorithm, a tool was built to visualize the results. This tool was developed as a web application in Python using Dash [1] as a frontend. There are four main plots that are the outputs of the tool:

- *Complexity animation* For every time step in the window of interest, the positions, interdependencies and complexity contributions for each aircraft are shown. This information is shown as an animation through the time window. The goal of this output is to clearly convey the evolution of complexity during a particular time window. In order to visually indicate when a community is complex, the interdependencies between aircraft of this community are colored in red.

- *Strength indicator animation* This plot provides similar information as the previous one. However, in this only the strength indicator is shown. More specifically, for each aircraft we provide the value of the maximal weight of the pairwise interdependencies that it is part of. As the strength indicator is defined through pairwise distances, it is directly linked to conflicts and losses of separation. Thus, the goal of this plot is to show specific safety related information.

- *Heatmap of complex communities* This output shows as a heatmap the contribution of every complex community that has existed through the duration of the time window. As the x-axis is time, the coexistence or any other time relation between complex communities can also be inferred. We also keep track of the aircraft in the sector that do not belong to a complex community, which we refer to as "Pool". All aircraft that are responsible for some of the complexity in the sector are shown there. These aircraft are also part of communities that are responsible for less than the complexity threshold. When no complex communities exist, the Pool is responsible for 100 % of complexity.

- *Summary table* This table shows a detailed summary of every complex community that has existed in the time window. We show relevant information

[1]`https://plotly.com/dash/`

such as start and end time, all members that have been part of the community and when each member was added and removed.

Lastly, there is also the possibility to generate and download a summary file for the current log file. This summary file contains the values of the input parameters, as well as statistical information regarding the number, size, duration and percentage of communities. The statistical information comprises of the mean, standard deviation and minimum and maximum values. However, such a functionality could be easily adapted or extended with respect to the needs of the practitioners.

The workflow to use the tool is shown in Figure 4.5. In this work, we use BlueSky [109] as the simulation platform for the trajectories. From BlueSky the tool requires as input a file that for every time step logs the positions of every aircraft. We note that it is not a requirement to use BlueSky and the tool is not dependent on it. The file with the logged information is, however, a requirement. Furthermore, there are three more inputs to the tool, namely the minimal and maximal thresholds in order to form the interdependencies and the complexity threshold for a community to be considered complex. As this tool has been written in Python, it will be straightforward to extend the functionalities of the tool and also integrate it with other existing tools, services and infrastructure. The tool has been open sourced[2].

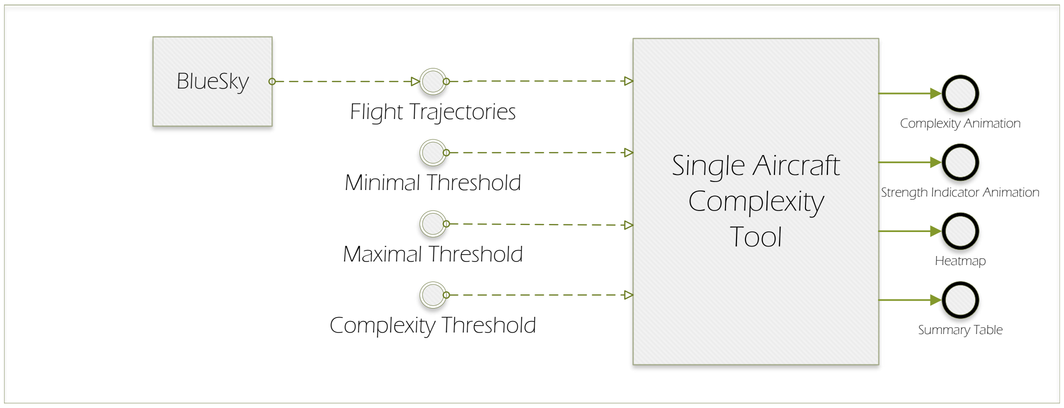


Figure 4.5: Workflow for visualization tool.

4.4 Results

Synthetic Traffic

In this section, we will describe several scenarios based on synthetic traffic to showcase how the information provided by the methodology and the tool can be

[2]https://github.com/risufaj/Single-Aircraft-Complexity

utilized. First of all, we will show a scenario where we analyze ATC decisions. In another scenario, we illustrate how ATFM decisions can affect KPAs.

Pairwise Conflicts

In this scenario, shown in Figure 4.6, we start with an arbitrary sector and 5 present aircraft and the simulation lasts for 15 minutes. The trajectories have been generated in such a way, that there will be a conflict between AC2-AC3 and AC4-AC5 at some time t. Furthermore, the conflict between AC4-AC5 starts slightly earlier, but both conflicts will co-exist in time.

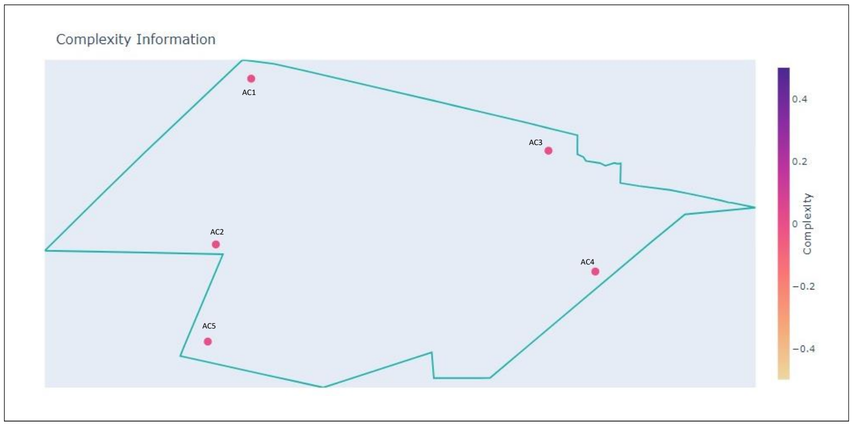
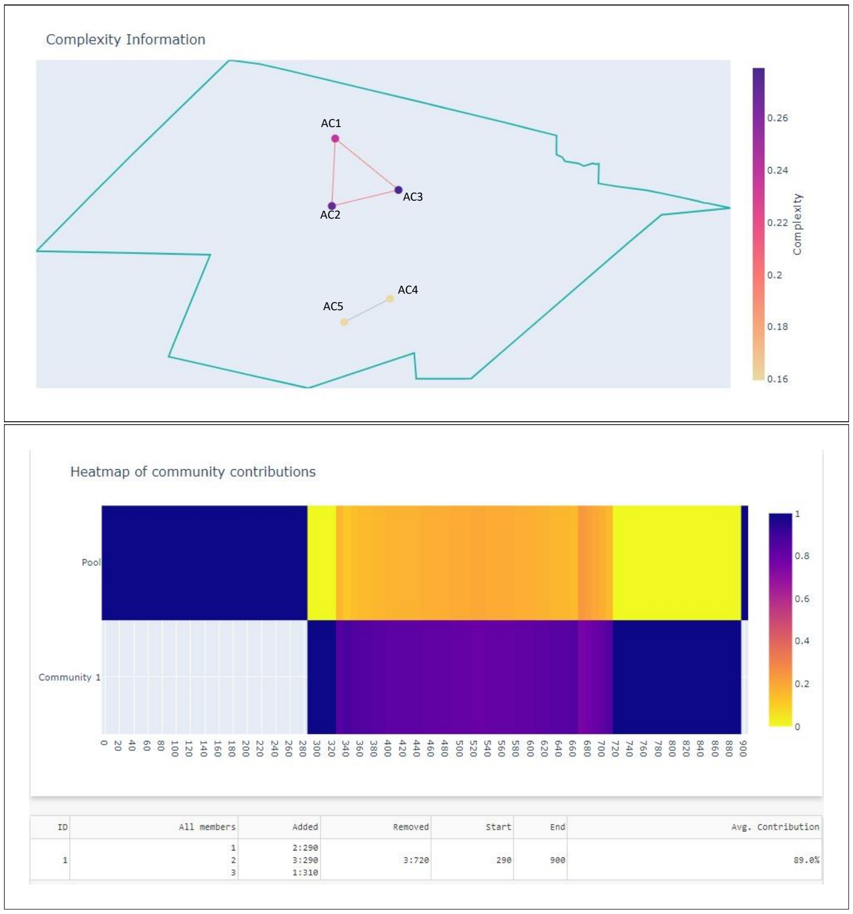


Figure 4.6: Initial state of scenario.

In such a situation, the ATCo will have to solve two conflicts that are happening around the same time and we assume that they are unable to solve them simultaneously. It is also assumed that there is enough time for ATCos to prevent losses of separation in pairwise conflicts.

In Figure 4.8 the state of the aircraft when the ATCo should have been alerted by the present Conflict Detection method is illustrated. This is shown in the tool through the Strength indicator, which directly correlates with the relative state between aircraft. The Strength of AC2, AC3 is 0.55 and AC4 and AC5 is around 0.6 (1 is a loss of separation). Typically, ATCos would solve the earlier conflict first, i.e, AC4-AC5. However, the presence of AC1 complicates this decision. Following the evolution of trajectories, AC1 will eventually create a compound conflict [110] with AC2 and AC3. This is not shown yet by using only the Strength indicator, however, when measuring single aircraft complexity as previously described, we can observe the following complexity situation shown in Figure 4.7. There, it can be seen that the community created by AC1, AC2 and AC3 is a complex community.



ID	All members	Added	Removed	Start	End	Avg. Contribution
1	1 2 3	2:290 3:290 1:310	3:720	290	900	89.0%

Figure 4.7: Complexity of aircraft.

The provided information should affect the decision of the ATC by considering three different KPAs.

- *Safety* - First of all, if the conflict between AC4 and AC5 is solved first, it is not guaranteed that it will be done before AC1 is in conflict with AC2 and AC3. This means that the ATCos would have to solve a compound conflict. The algorithm quantifies this information and the tool presents it in such a way that clearly illustrates which aircraft form the compound conflicts and complex communities. Therefore, using the information provided by the tool, the ATCo should make the decision of to solve the pairwise conflict between AC2 and AC3 first. Furthermore, the conflicts could be resolved in such a way that at best reduces the overall complexity and at worst just avoids secondary conflicts. This information could be acquired by running the algorithm again after a resolution is proposed.

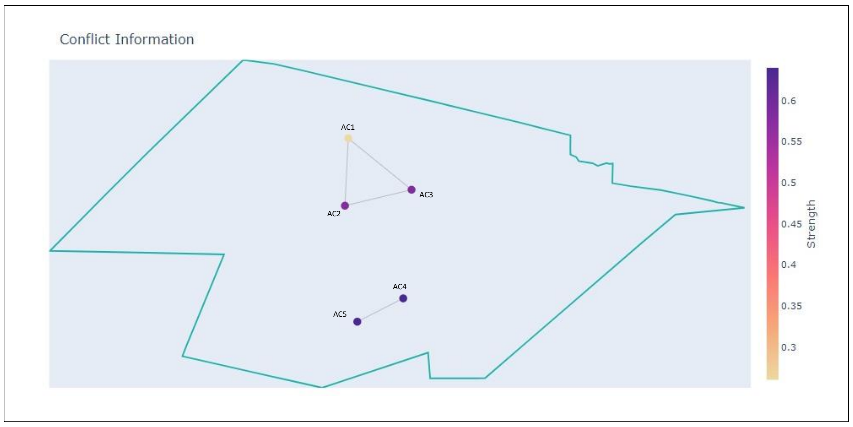


Figure 4.8: Conflict State.

- *Efficiency* - However, the controllers might still be able to solve the compound conflict. One way to solve it could be to force one of the aircraft to have a large deviation from its original trajectory. This solution effectively would reduce the compound conflict to a pairwise conflict. However However such a resolution would not be preferred as the aircraft that is deviated will incur delays that result in inefficient use of time and fuel.

- *Capacity* - Nevertheless, delays to one of the aircraft might be unavoidable. Another option could for the controller to determine that they are not able to solve these conflicts in time. Let us assume that in this case, ATC would make a request for one of these aircraft to be delayed. The ATCos will have the information that which aircraft are in conflict and which aircraft form a complex community. Consequently, delaying one of the aircraft could be done by maintaining some fairness and therefore one of the aircraft of the complex community should be delayed. Determining which of them would depend on the capabilities of the ATCos to solve two pairwise conflicts around the same time.

Deconstructing Complex Communities

In this scenario, we are studying the same sector as in the previous section. There will be 7 aircraft present in total through the simulation which lasts 20 minutes. AC1 is present throughout the simulation and forms various interdependencies with the other aircraft. The other aircraft were generated in such a way that they intersect with the trajectory of AC1 at different time steps. There were no restrictions on whether the other aircraft can form interdependencies amongst themselves. This is illustrated in Figures 4.9, 4.10 and 4.11 where the sector is shown in three different time steps; initial t_0 where AC1 has interdependencies with AC2 and AC3 but not with AC4 and AC5.

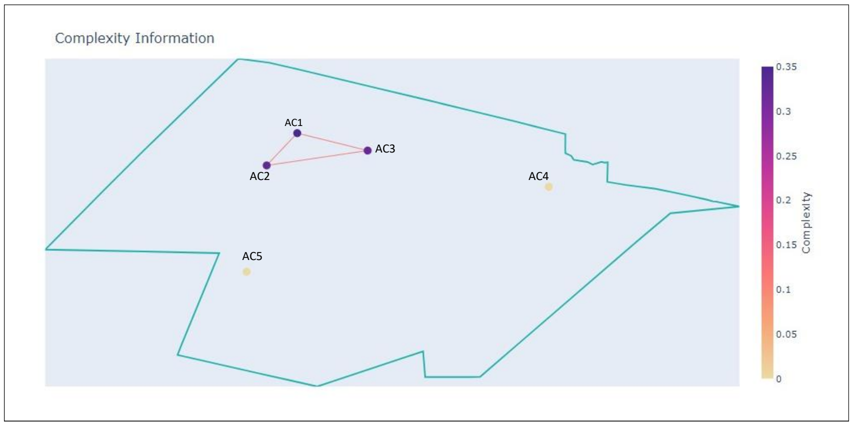


Figure 4.9: Initial state of the sector t_0.

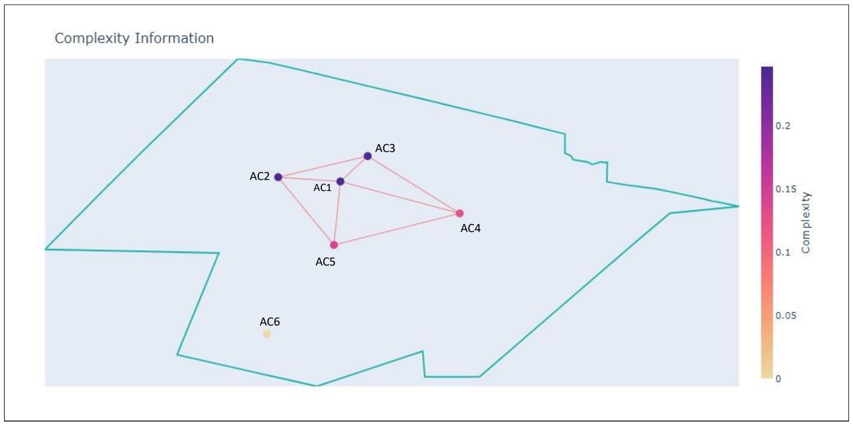


Figure 4.10: Sector in t_1.

In t_1 where AC1 has created interdependencies with AC4 and AC5, while the remaining aircraft have also created interdendencies amongst themselves. In t_2 the evolution when AC1 has travelled through the sector enough to create interdependencies with AC6 and AC7.

The complexity evolution is shown in Figure 4.12. As it can be seen, one complex community is detected that lasts from $t = 130$s to $t = 1200$s and is comprised of all the members that have passed through the sector during the simulation time. The community starts with AC1, AC2 and AC3 and during the simulation, AC1 forms interdependencies with AC4-AC7 which causes the initial community to evolve.

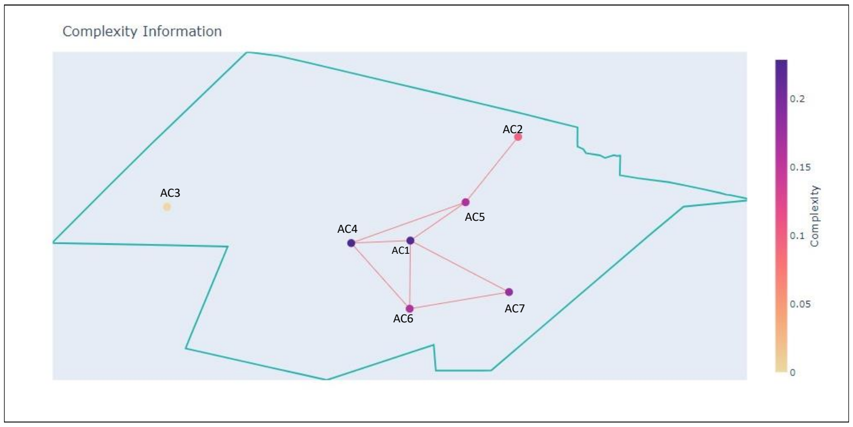
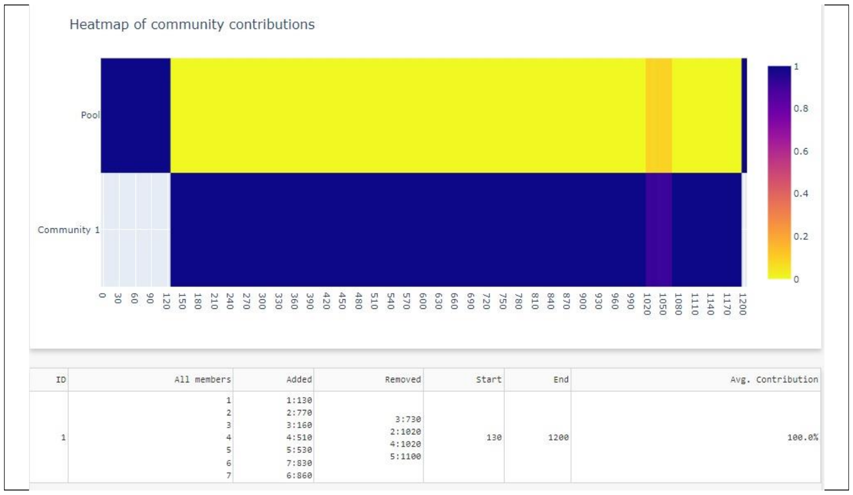


Figure 4.11: Sector in t_2.

According to Algorithm 2 communities in consequent time steps will be considered the same if they are similar enough. In this case, from Figure 4.10 we observe that the community is comprised of AC1 - AC5 while in Figure 4.11 the community is comprised of AC1-AC7 with the exception of AC3. Thus, AC1 is a permanent member of the complex community which causes the community to last so long.



ID	All members	Added	Removed	Start	End	Avg. Contribution
1	1 2 3 4 5 6 7	1:130 2:770 3:160 4:510 5:530 7:830 6:860	3:730 2:1020 4:1020 5:1100	130	1200	100.0%

Figure 4.12: Complexity evolution.

The situation illustrated above could be problematic for the ATC as it demands its continous attention throughout the time window as a result of the nature of the interdependencies. Therefore, it is reasonable to expect that the ATM system should intervene to make this situation more manageable. The information that the algorithm provides through the tool could be used to better formalize and understand the consequences of the decisions. For instance, let us assume the absence of AC1 in the sector during the window of the simulation (e.g., delay, flight plan change). This is illustrated in Figures 4.13, 4.14 and 4.15. Furthermore, the complexity evolution in this case is shown in 4.16.

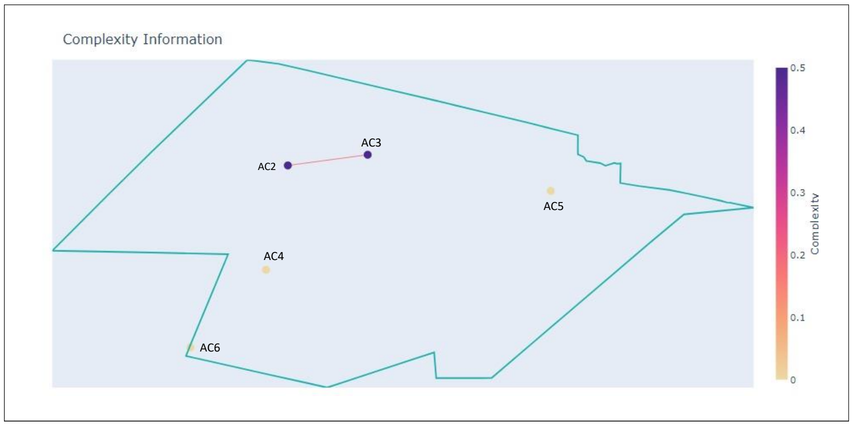


Figure 4.13: Initial state of the sector t_0 without AC1.

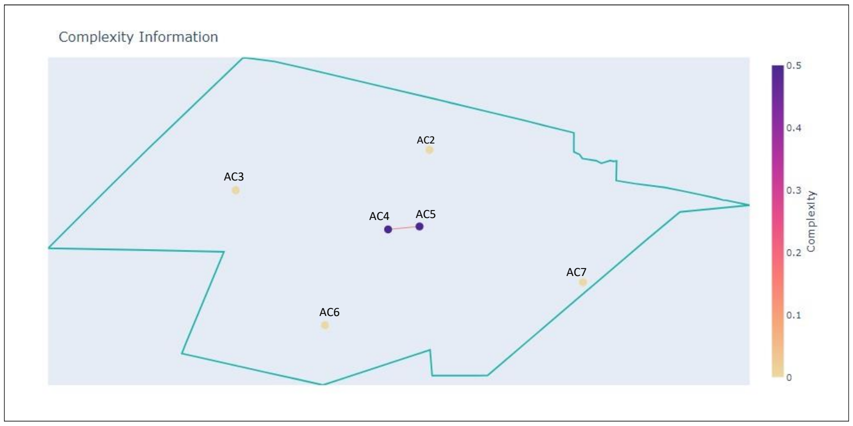


Figure 4.14: Sector in t_1 without AC1.

In this situation the absence of AC1 has resulted in the previous complex community to be split in 3 smaller but still complex communities. It can also be noted that these communities do not overlap in time. The communities have only 2 members which causes only Strength to be a non-zero complexity indicator. This situation is less complex than the previous one where several different topologies were present in the sector.

The lack of understanding of how and why the complex situation arises, may lead the different ATM subsystems to make arbitrary decisions which can clearly affect efficiency and fairness. Let us assume that both ATFM and ATC agree that some sort

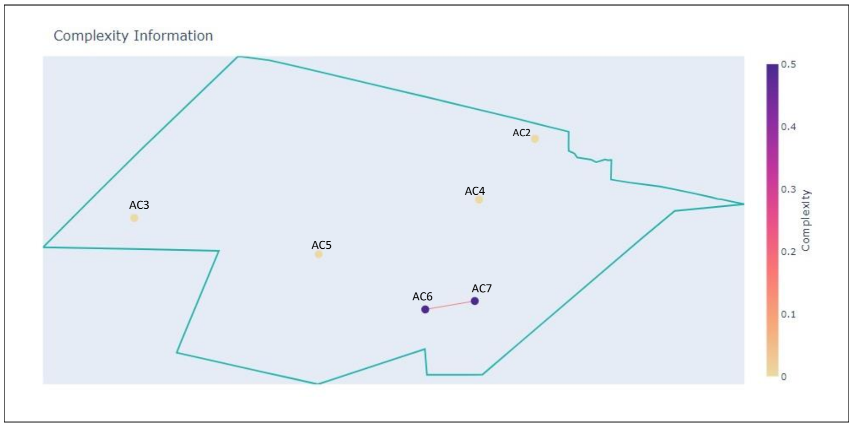
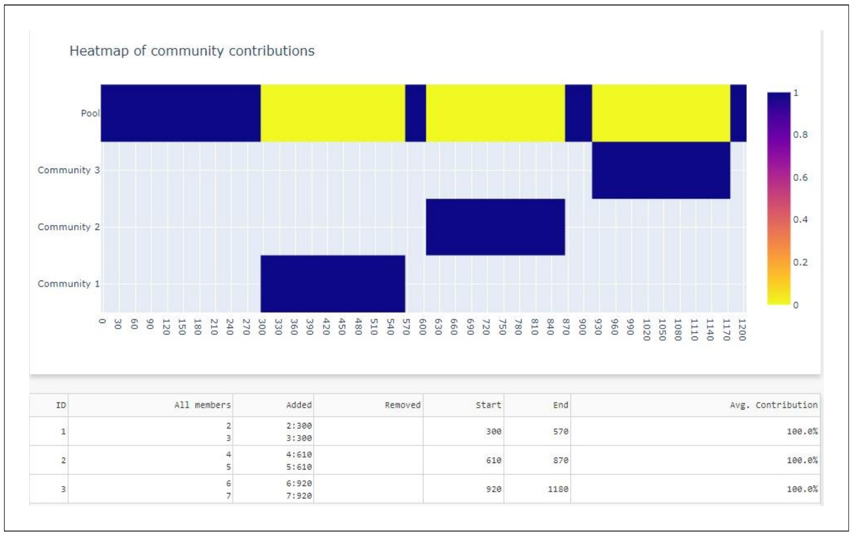


Figure 4.15: Sector in t_2 without AC1.



ID	All members	Added	Removed	Start	End	Avg. Contribution
1	2 3	2:300 3:300		300	570	100.0%
2	4 5	4:610 5:610		610	870	100.0%
3	6 7	6:920 7:920		920	1180	100.0%

Figure 4.16: Complexity evolution without AC1.

of regulation needs to be made. However, each of them could propose regulations that affect one or more aircraft. Using the information provided from the tool, the aircraft that needs to be affected from the regulations becomes evident. As previously stated, the absence of AC1 results in a more manageable situation. This is evidence that the use of the methodology proposed in this work leads to better equity and fairness at the aircraft or airline level. The objectiveness of the provided information results in a neutral tool that increases transparency and explainability of decisions made by ATM subsystems with regards to AUs.

Flown Trajectories

Data

We will evaluate the algorithm using real historical traffic from 17.08.2019 provided by CRIDA. The available data contained flown trajectories in several sectors over Spain starting from noon and lasting for about 7 hours. In total there were 485 flights in the dataset. Furthermore, the dataset contained a list of flights that were regulated for the Pamplona Upper ATC Sector (LECMPAU), shown in Figure 4.17. The type of regulations were time delays issued from ATC due to airspace capacity. Preliminary analysis of the data showed that ouf the 485 flights, 329 had crossed the LECMPAU sector and 82 out of those were regulated (24.9 %). The mean delay was 14.8 minutes with a standard deviation of 11.2 minutes. The minimum delay was 1 minute while the maximum was 59 minutes.

Using the information available, we simulated two scenarios in Bluesky, one with the applied regulations and the other without the applied regulations.

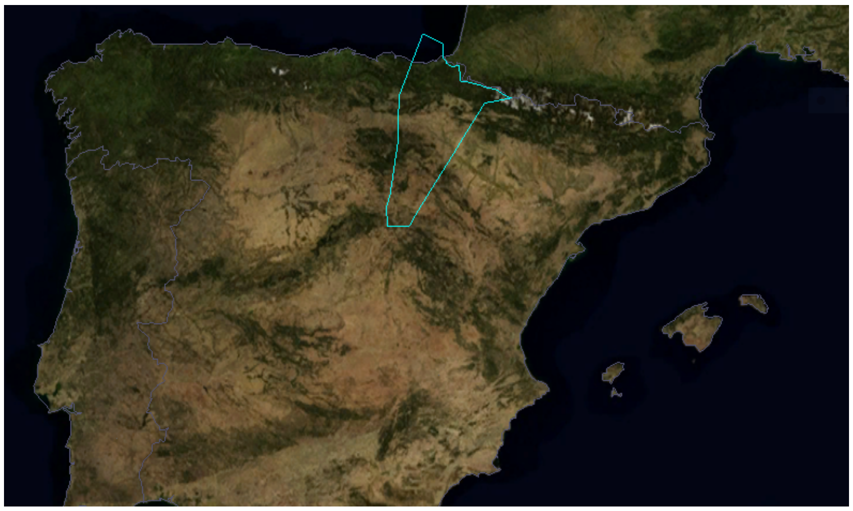


Figure 4.17: Pamplona Upper ATC Sector.

The Effect of Regulations on Complex Communities

In this section we will inspect how complex communities are affected by the regulations present in the data set. In order to do so, we simulate the two scenarios for the total duration of 7 hours. The trajectories with the regulations applied are the actual flown trajectories. However, we are not aware of how and when the delays were applied exactly. Thus, in order to create the trajectories without the regulations applied, we always remove the delay from the initial point of the trajectory.

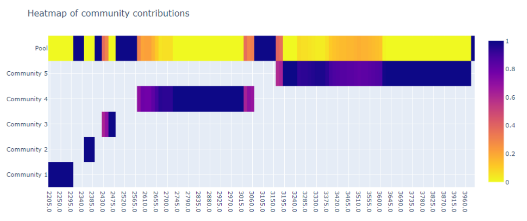


Figure 4.18: Complex communities for the first time window with regulations.

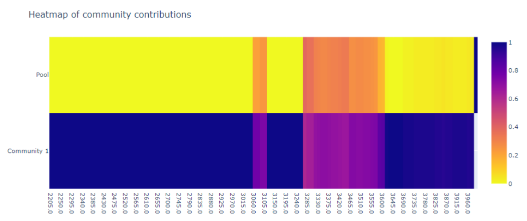


Figure 4.19: Complex communities for the first time window without regulations.

In order to visualize the effects, we identified three windows lasting 30 minutes each. The input parameters for the minimal and maximal distance thresholds and the complexity thresholds were set to 5 NM, 33 NM and 60% respectively. The complex communities for the first time window are shown in Figures 4.18 and 4.19. As it can be observed, in the scenario where the regulations were not applied there exists only one community throughout the time window. Compared to the scenario where the regulations were applied, we observe 5 total complex communities. As a result of the complexity threshold being set to 60%, these communities do not co-exist in time. This result provides evidence that the delayed aircraft were key in keeping the community in Figure 4.19 together, similar to the synthetic scenario elaborated in Section 4.4. The presence of only one complex community indicates that all relevant aircraft in the sector have interdependencies with them, which is also suggested by the colour of the heatmap, with the community being responsible for 100% of the complexity for the majority of the time window. This topology is evidence of a very complex situation. Through the regulations, we can observe that the complexity is divided in time between several communities. Communities 1,2 and 3 have a relatively short duration, however they are responsible for all of the complexity in the sector. This suggests that the controller is provided elaborated information about which areas of the sector are causing the complexity present in the sector. Such information is further supported by the other outputs of the tool, e.g., the animation of single aircraft complexity contributions. This scenario serves to further illustrate two advantages of the proposed methodology: it increases the transparency of decisions made by different sub-systems of ATM and provides a framework through which to formalize the need for regulations (in this case from the point of view of ATC).

However, as it can be seen in the second (Figures 4.20 and 4.21) and third (Figures 4.22 and 4.23) time windows, this is not the full story. There, we can observe that the

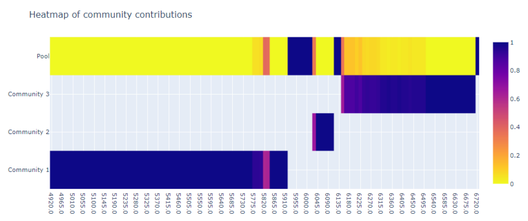


Figure 4.20: Complex communities for the second time window with regulations.

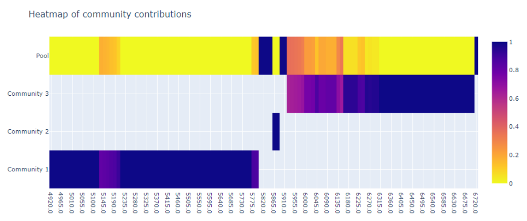


Figure 4.21: Complex communities for the second time window without regulations.

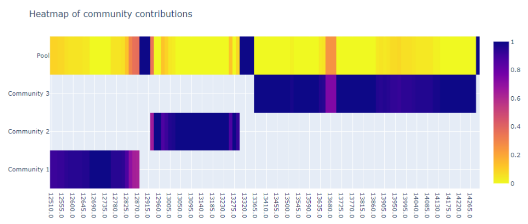


Figure 4.22: Complex communities for the third time window with regulations.

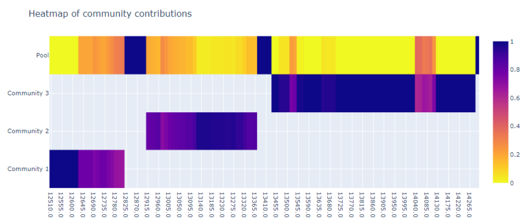


Figure 4.23: Complex communities for the third time window without regulations.

regulations change the composition of complex communities only slightly. These results suggest that complexity largely remains the same in both scenarios (i.e., with and without regulations applied). Furthermore, it is interesting to note that during these time window the difference in occupancy between the two scenarios was the highest. In the scenario without the applied regulations there were consistently 5-7 more aircraft than in the one with the regulation applied. Such a result echoes claims made in other works [104, 41] that the occupancy of sectors does not provide elaborated information about the complexity of the traffic.

Moreover, it is worth noticing that in the scenarios without the regulations applied, the Pool tends to be responsible for more complexity. This means that the delayed aircraft were largely not present in the complex communities, but in the surrounding traffic. Such a distribution of complexity is evidence of the granularity of the information provided by the proposed algorithm, as it is able to capture subtle differences in sector complexity, which could explain in a nuanced way why the particular regulations were applied, something that sector occupancy cannot provide.

Sensitivity Analysis

The algorithm proposed in this work has 3 main parameters: minimal and maximal interdependency thresholds and the threshold for a community to be considered complex. These parameters should depend on the problem setting and also the individual user, however it is important to understand how they affect the output of the algorithm. We conduct a sensitivity analysis, which investigates how the output of a system can be attributed to its inputs. More specifically, we use the Sobol method [111, 112] which is a variance-based global sensitivity analysis. The variance of the output is decomposed into fractions which are attributed to the inputs. The main advantage of using this method lies in the fact that it deals with nonlinear responses and it can measure the interactions between input parameters. In order to perform a Sobol analysis, a parameter sequence is generated, which in

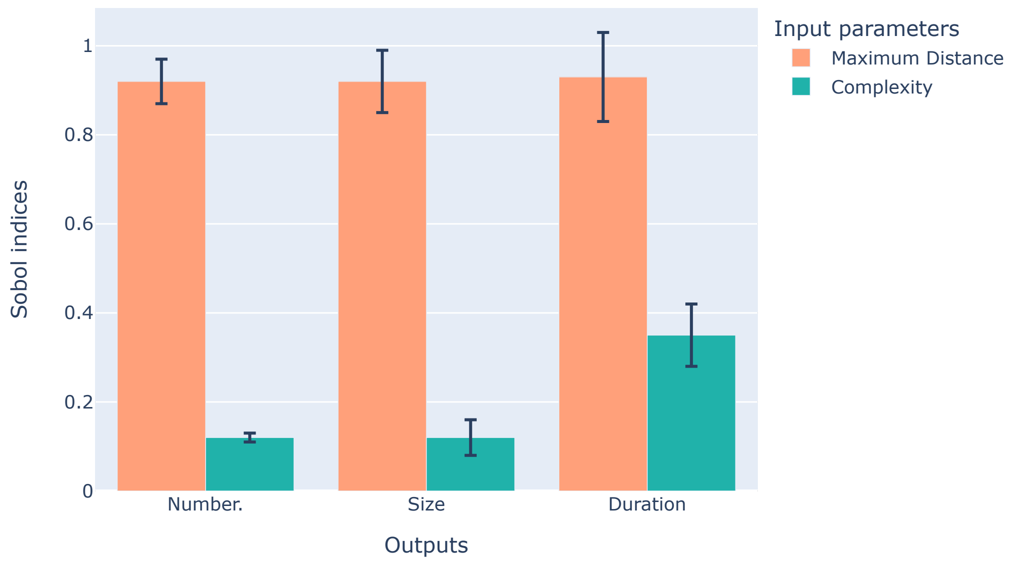


Figure 4.24: The Total Sensitivity Analysis using the Sobol Method .

this work returns a Sobol sequence using Saltelli's sampling scheme [112]. A Sobol sequence is a quasi-randomized, low-discrepancy sequence that samples the space more uniformly than a completely random sequence. The Saltelli scheme extends this sequence in a way that reduces the error rates in the calculations. To understand how the variance of the output can be attributed to the input tarameters and the interaction between each of them, the total order, first and second order sensitivity indices are calculated. The first order sensitivity indices are used to measure the fractional contribution of a single parameter to the output. Second order sensitivity anaylsis are used to measure the contribution of parameter interactions to the output variance. The total sensitivity indices take into account all the previous indices.

In this work, the sensitiviy analysis will be conducted using the regulated data described in Section 4.4. In order to reduce the computational burden of the analysis, motivated furthermore by the fact that this data has been of particular importance for the ATC, we keep a fixed minimal distance threshold of 5 NM. The ranges for the maximal distance threshold and the complexity threshold were $[15, 75]$ NM and $[40\%, 100\%]$ respectively. To perform the analysis 6200 different combinations were generated. Lastly, measured the response of three different outputs of the algorithm: number of communities, median size of communities (number of total members) and median duration of the communities.

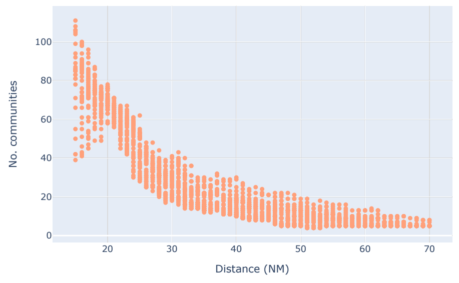


Figure 4.25: Number of communities for different distance thresholds.

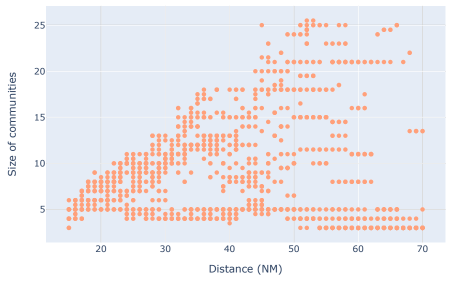


Figure 4.26: Size of communities for different distance thresholds.

The results of the sensitivity analysis are shown in Figure 4.24. As it can be observed, for all outputs, the total sensitivity Sobol indices indicate that the the maximal distance threshold for the interdependencies is the input that affects the output the most. In this work, the graph and in turn the communities are generated by considering the distance threshold in order to build the edges. Thus, it is evident that this should affect the number of communities and size of communities the most. For instance, when the distance is large enough, then at time t it is reasonable to expect that the traffic graph is fully connected. In such a scenario, the complexity threshold would be irrelevant, as the community induced by the fully connected graph is always responsible for 100% of the complexity at time t. A similar example could be used also to illustrate why this happens for the size of the communities. A large (or small) enough distance threshold will largely dictate which aircraft form interdependencies. In either extreme value, it is clear to see that the relevant aircraft would be forming the complex communities, thus determining also the size of the communities. Figures 4.25 and 4.26 are further evidence of this. As it can be seen, the number and size of communities is strongly dependent on the distance threshold. The number of communities is inversely proportional with the threshold, while the size of the communities is proportional. We also observe a correlation between the size of communities with the complexity threshold, shown in Figure 4.27. Nevertheless, as the sensitivity analysis suggests, this is as a result of the correlation between the two input parameters. In order for a community to be responsible for 100% of the complexity, it should contain all aircraft that have at least one interdependency, which can be the case for bigger distance thresholds.

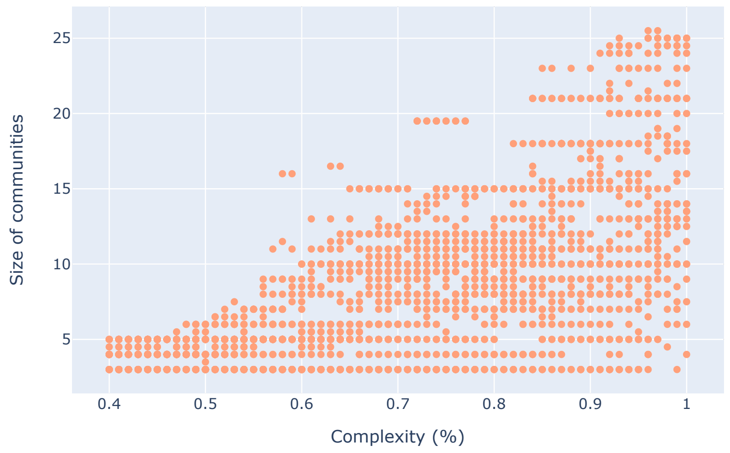


Figure 4.27: Size of communities for different complexity thresholds.

From Figure 4.24, it can be observed that the median duration is also affected mostly from the distance threshold, further evidenced in Figure 4.28. There, we can see that when the distance threshold is low, communities tend to last less. This happens because smaller distance thresholds are very sensitive towards the changes in the aircraft positions, as evidenced also from Figure 4.25, where smaller distance thresholds lead to many present communities.

To give some insight on how the median duration is affected by the complexity

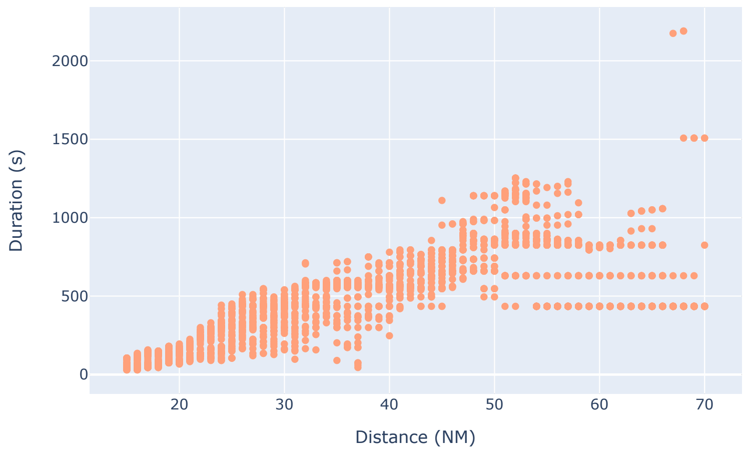


Figure 4.28: Duration of communities for different distance thresholds.

threshold, we show Figures 4.29 and 4.30. When the distance threshold is less than 25 NM, we observe that the duration decreases with increased complexity threshold. Such a result can be explained by the fact that with such a small distance threshold, it is more likely to have communities that are comprised of a subset of the aircraft co-existing in time in the sector. Consequently, aircraft can join communities and allow for communities to exist for longer. Nevertheless, this interpretation does not tell the whole story, as it can be observed that in such a setting the longest community existed for around 400s. On the other hand, when the distance threshold is bigger than 25 NM, the duration of communities is affected less by the complexity threshold. In this setting, we can observe that most communities last less than 17 minutes (less than 1000 s). The different response for different distance thresholds explains the Sobol index for the complexity threshold in Figure 4.24.

While we show that the algorithm is mostly affected by the distance threshold, this does not indicate what input values different ATM subsystems should use. In fact, the analysis that we conducted in this section should serve for practitioners (e.g., NM, ATC etc.) to determine what values are more suitable for their use-case. For instance, a similar analysis could be used as a baseline to quantify controller preferences, whether that be in the topology of the graph the traffic induces, or any of the other outputs provided by the propsoed algorithm in this paper. Finally, while we do not expect the behavior of the algorithm to fundamentally change with other

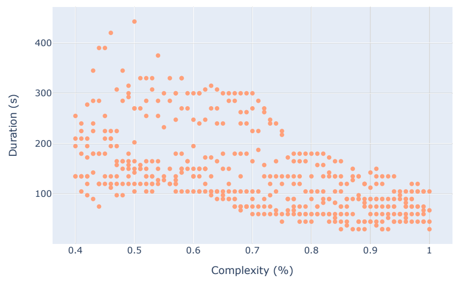


Figure 4.29: Duration of communities for different complexity thresholds (Distance threshold < 25 NM).

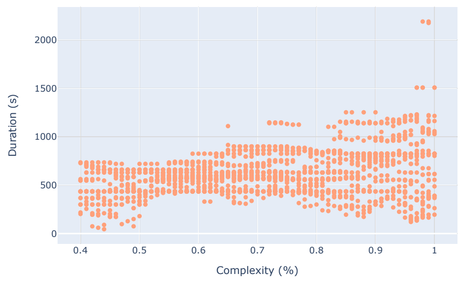


Figure 4.30: Duration of communities for different complexity thresholds (Distance threshold > 25 NM).

datasets, the analysis conducted in this section also heavily depends on the sector (or any other division of airspace) that is being investigated.

4.5 Conclusions

In this work, we propose a methodology that extends existing air traffic complexity indicators based on dynamic graphs to provide highly granular and nuanced information. As such, the concept of *single aircraft complexity* is proposed which measures the individual contribution of each aircraft to the overall sector complexity. Furthermore, the algorithm provides *complex communities*, which are connected components of the air traffic graph in order to determine complex spatio-temporal areas in the sector.

To effectively illustrate the algorithm, a web application was developed which visualizes several outputs of the algorithm, namely: a complexity animation, a strength indicator animation, a heatmap of complex communities and a summary table of complex communities. Furthermore, the tool also provides the user with the possibility to download a summary file for the scenario being investigated. The tool (and the underlying algorithm) are envisioned as neutral aids that can ease the smooth functional transition between ATM layers and DSS tools that should be used in union with existing tools. Furthermore, the information provided could enhance equity, fairness at the aircraft of airline granularity level.

In order to support our claims, we provide detailed use cases based on synthetic traffic, as well as real historical traffic. We first show that the algorithm can serve to formalize controller decisions as well as guide controllers to better decisions in situations where multiple pairwise conflicts co-exist in time. Further, we investigate how the provided information can be used to increase transparency of the decision makers towards different AUs, which serves also to increase fairness and equity. Moreover, the algorithm was evaluated using historical traffic, which was regulated through delays to several aircraft as a result of ATC capacity. We constructed two scenarios: one with the regulations applied and the other without the regulations applied and showed how the complex communities were affected in three 30 minute time windows. Finally, an extensive sensitivity analysis for two of the inputs to the algorithm (maximum distance threshold to form interdependencies and the complexity threshold for a community to be considered complex) was conducted. The sensitivity analysis was conducted for three outputs of the system: number of communities, median size of communities and median duration communities. We found that the maximum distance threshold affected these outputs the most. To fully understand this result, the response of each output to the different input values was studied. We argued how a similar analysis could be used to quantify controller preferences for graph topologies in the sector.

Nevertheless, the proposed algorithm should be extended and further refined. Most importantly, as one of the inputs to the tool are trajectories in time, a way to consider uncertainties should be investigated. As previously mentioned, the tool is envisioned to be used alongside existing tools, therefore one way to consider uncertainty would be for the inputs to the tool to have modelled uncertainty beforehand. However, it could be interesting to extend the definition of the graph to contain not a weight for any interdependency, but a distribution of weights.

Moreover, the input parameters of the algorithm are envisioned to be tuned according to the problem and preferences of the practitioners. Nevertheless, the tool could have several "modes" when given a certain value for the distance and complexity thresholds. For instance, a more conservative mode could involve considering larger inputs that would lead to bigger communities that last longer, but visually provide more information about aircraft that are further in the community. Through the animation output, the practitioner could still visualize the core of the community. A less conservative mode would instead only consider the core of the communities, in order to provide only crucial members of the communities.

The information provided should be evaluated by practitioners in a user study in order to optimize how and what information is shown. Finally, the tool should be further developed into a mature DSS, in order to be used alongside existing tools and methodologies.

Chapter 5

TOWARDS CONFLICT RESOLUTION WITH MULTI-AGENT REINFORCEMENT LEARNING

Safety in Air Traffic Management at the tactical level is ensured by human controllers. Automatic Detection and Resolution 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 to improve the quality of resolutions based on a combination of several factors. We use Multi-Agent Deep Deterministic Policy Gradient 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.

5.1 Introduction

The mission of air traffic management (ATM) is to make air traffic possible by means of efficient, environmentally friendly and socially valuable systems [69, 70]. 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 [71].

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 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 [113]. Early works, such as [114] 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. [115] 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 [116] 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 displace 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. [75] 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 [117] 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.[76]. 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 [118] and Go [119], as well as impressive performance in practical problems such as autonomous driving [120]. 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.[57] propose a similar approach. However, they do not consider a multi-agent environment and have a less specialized reward function. Ribeiro et al. [121] 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) [122] 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 airspace complexity. 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 (i.e. flown trajectories), with a data augmentation technique to increase the variance of employed scenarios. Each scenario lasts 20 minutes, which is a common length for algorithms in the tactical level [75, 57]. Agents act on a 15-second time step and are able to solve the majority of conflicts in only 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 $[v - 6\%, v + 3\%]$ [76].

The rest of this paper is organized as follows: in Section 5.2, we elaborate on the theoretical background necessary for this paper. In Section 5.3, the experimental setup is presented. Results are presented and discussed in Section 5.4, while in Section 5.5 we draw conclusions and propose steps for further research.

5.2 Theoretical Background

Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of machine learning which deals with sequential decision making[123]. 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 [124].

A given RL problem is usually formalized by a Markov Decision Process (MDP), which is a discrete time stochastic control process [125] that consists of a 4-tuple (S, A, T, R), where:

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

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5.1}$$

where γ is a discount factor between 0 and 1 which informs the agent how relevant immediate rewards are in relation to future rewards.

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.

The goal of a RL algorithm is to find a policy which maximizes the total future discounted reward. There are two common ways that are used to do this: the value function V^π and the action-value function Q^π which are defined as follows:

$$V^\pi(s) = \mathbb{E}_\pi(R_t | s_t = s) \tag{5.2}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi(R_t | s_t = s, a_t = a) \tag{5.3}$$

The value function quantifies the future expected reward in the current state if the policy π is followed. On the other hand, the Q action-value function represents the expected rewards for the state-action pairs if policy π is followed.

The most notable algorithm for solving MDPs (and in turn the RL problem) is Q-learning. In Q-learning, an agent learns to estimate the optimal action-value function which is represented as a table with as many state-action pair entries as possible[123]. However, this is not feasible for a number of real life applications, where the state and action spaces are large or continuous. As a result there can be an infinite number of state-action pairs. In such instances, the Q-function must be approximated by another function. Typically, the approximator function with parameters μ is optimized through the Bellman equation [125].

Deep Q-Networks (DQN)[118] uses neural networks as Q function approximators. Nevertheless, the original implementation of DQN had several issues, which have been adressed in several ways. To begin with, the rewards in a RL problem can be sparse or delayed. This is a problem for neural networks as they rely on directly gained feedback. Furthermore, the data in a RL problem are typically highly correlated as they come from continuous interactions with the environment. Also, as the policy changes, the distribution of the data changes as the agent interacts differently with the environment (i.e. takes a different action for the same state which can lead to a previously unseen state). As a result, the data becomes non-stationary which typically makes the training of neural networks unstable. Consequentially undesired phenomena such as *catastrophic forgetting*, where the agent suddenly cannot solve the task anymore after apparently learning an adequate policy.

The non-stationarity issue is usually solved using *target networks*, which are identical neural networks to the ones used for learning the Q-function that are held constant for several training steps to serve as a stable training target. Sample correlation has been solved using *experience replay* [118]. This technique stores the agent's experience at each time step in a replay buffer. Random samples from the memory are used to update the networks. When the buffer becomes full several methods can be employed, however the simplest (and often the most successful one by empirical evidence) is to discard the oldest samples.

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 [126], which are decision processes for N agents.

Similar 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.

Multi-Agent Deep Deterministic Policy Gradient

As mentioned, Q-learning and DQN attempt to maximize the expected value of the total reward for a given and all successive steps [123, 118]. However, it has been noted that this method often suffers in high dimensional action and state spaces and can fail to converge [127, 122].

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:

$$\theta_{i+1} = \theta + \alpha \triangledown J(\theta_i) \tag{5.4}$$

where θ are the policy weights, α is the step size and $J(\theta)$ is a loss function to measure how well the model is performing. Usually, the loss function uses the value function with the policy estimation to take the action. 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 [123] 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 [127]. 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 algortihms that learn a deterministic target policy from trajectories that have been generated by a stochastic policy (we refer the reader to [128] for a more complete explanation).

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:

$$\pi' = \pi(s|\theta^{\pi}) + \mathcal{N} \tag{5.5}$$

where π is the policy, θ are the policy's parameters and $\mathcal{N}$ is an arbitrary noise distribution.

In this work we will use *Multi-Agent Deep Deterministic Policy Gradient* (MADDPG)[122], which is an extension of single agent DDPG[129], 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 [123, 126].

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.

5.3 Experimental Setup

Data and Parameters Used

The model is evaluated using traffic data from Eurocontrol's DDR II which contains high fidelity data of flown trajectories, from 12.02.2019 [130]. Conflicts were detected using a simple state based method [10]. 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. As we are only considering en-route traffic, filter was used to discard conflicts below FL250. Furthermore, except the conflict pair, we also keep several surrounding aircraft. A scenario contains a pairwise conflict and some surrounding traffic, which is an assumption we make for computational feasbility. Each agent is assigned to one of the conflicting aircraft. The trajectories of the present aircraft were projected in the future using a lookahead time of 300s, assuming constant speed and heading. The agents can only observe the 5 closest aircraft (including ownship). We utilize the method proposed in [78, 79] to identify relevant aircraft. The algorithm is described in Algorithm 4. The functions to identify conflicting and non-conflicting aircraft, as well as creating a conflicting aircraft for some given settings were taken from Bluesky [109]. This procedure resulted in a total of 188 conflict scenarios.

Algorithm 3 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
```

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 (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.

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.

State Representation

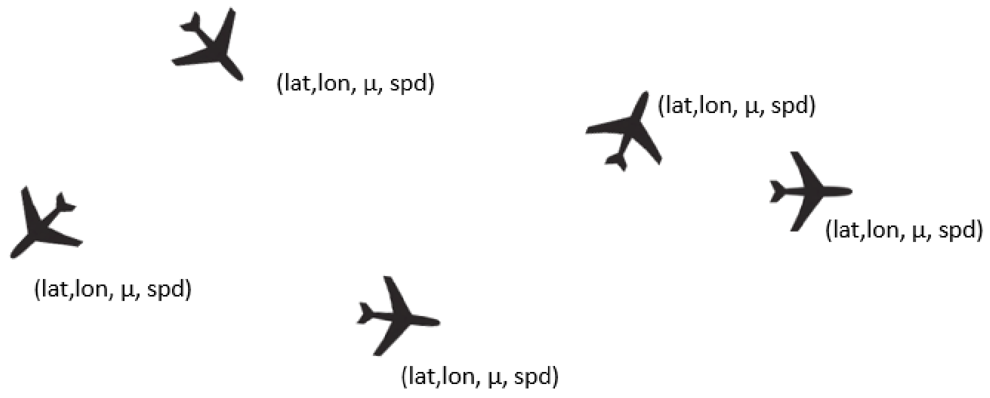


Figure 5.1: Position information of each aircraft in a scenario required to represent the state.

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 [121], 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 5.1, 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.

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 [127, 131, 122, 132, 133, 134]. Ribeiro et al. [121] and Pham et al. [57] use a reward function solely based on the number of conflicts and number of losses of separation 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:

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.

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.

Fuel consumption

In order to discourage the model from taking actions that lead to big fuel consumption, we use the aircraft performance model OpenAP [135]. As per this model, the aircraft receives a negative reward for the amount of fuel it consumes each time step.

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.

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 [31, 73]. While we implicitly consider complexity by measuring if the resolution causes new conflicts, this alone is not informative enough in terms of complexity

[31]. Complexity has been defined in several ways. The simplest and most used of them is aircraft density,defined by [26] as the number of aircraft co-existing in the airspace that is being studied. This however, has been shown not to give a full picture of complexity [79]. Furthermore, in this paper scenarios always have the same number of aircraft, which would result in the same complexity for all scenarios if we were to use this metric.

In this work, we consider the complexity formulation of Koca et al. [79]. They propose a hierarchical ecosystem structure, where relevant aircraft to the conflict are determined based on spatio-temporal interdependencies. Such interdependencies are defined on space-time regions, which in turn are areas where an aircraft can be conflict-free. If the regions of two aircraft intersect, then they are interdependent [79]. Let us assume that there is a conflict between aircraft. The set of aircraft involved in this conflict is denoted by C and the hierarchy over this traffic is denoted by H_C. The members of C form the members of the first order of H_C. Members of the i^{th} order are the aircraft that are not members of a lower order, but have an interdependency with a member of the $(i-1)^{th}$ order. Formally, this is defined:

$$\begin{cases} H_C(1) = C \\ H_C(i) = AC \in F | AC \in H_C^-(i) \land (\exists AC' \in H_C(i-1) : AC \perp\!\!\!\perp AC') \end{cases} \tag{5.6}$$

where F is the set of all aircraft to consider, $H_C^-(i) = F \setminus \bigcup_{j=1}^{j-1} H_C(j)$ and $\perp\!\!\!\perp$ denotes interdependency.

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 to collect as much reward and solving the conflict in an inefficient way [121]. The final reward is a weighted sum of all the single factors mentioned above.

The Model

In this work, we train two agents that represent each aircraft of the conflict pair. Figure 5.2 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

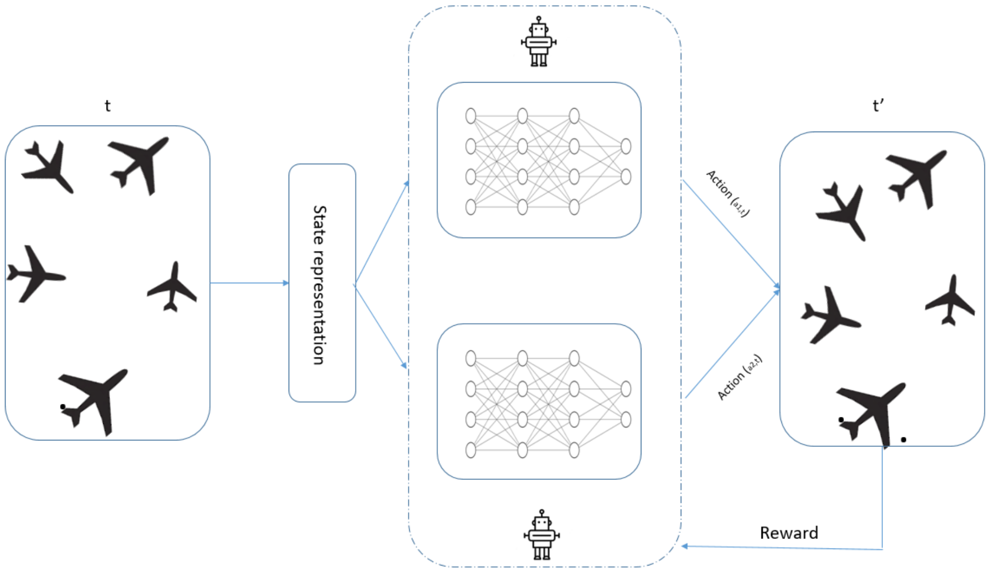


Figure 5.2: Model for conflict resolution.

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 ±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 $[v - 6\%, v + 3\%]$ [76, 136].

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.

Simulation Environment

Simulations were run on the air traffic simulator BlueSky [109]. 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.

5.4 Simulation Results

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.

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 5.3 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 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 behaviors are not preferred, it is encouraging to see that the majority of conflicts are solved in less than 5 time steps. This means

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

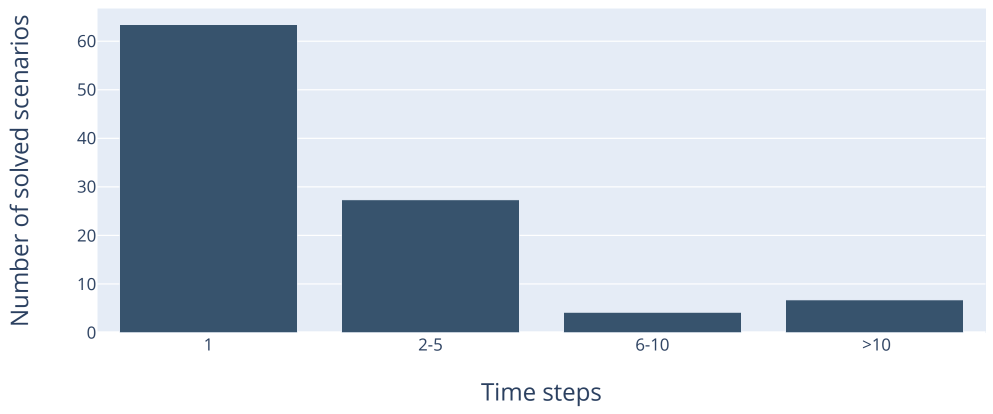


Figure 5.3: Number of steps required to solve the conflict. A time step is 15 seconds long.

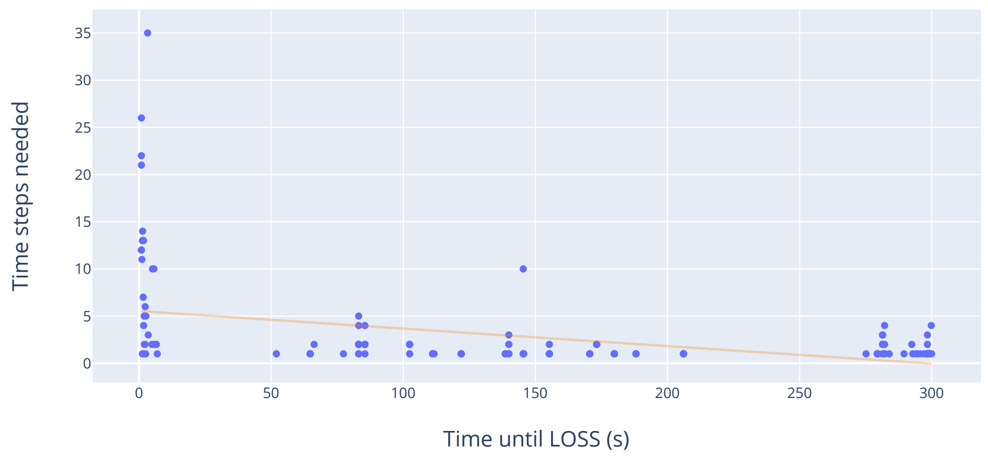


Figure 5.4: Relation between number of steps needed to solve the conflict and time before conflict starts.

that the majority of conflicts are solved in around 1 minute from when the conflict was detected.

Figure 5.4 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 result indicates moderate negative correlation, which can also be seen

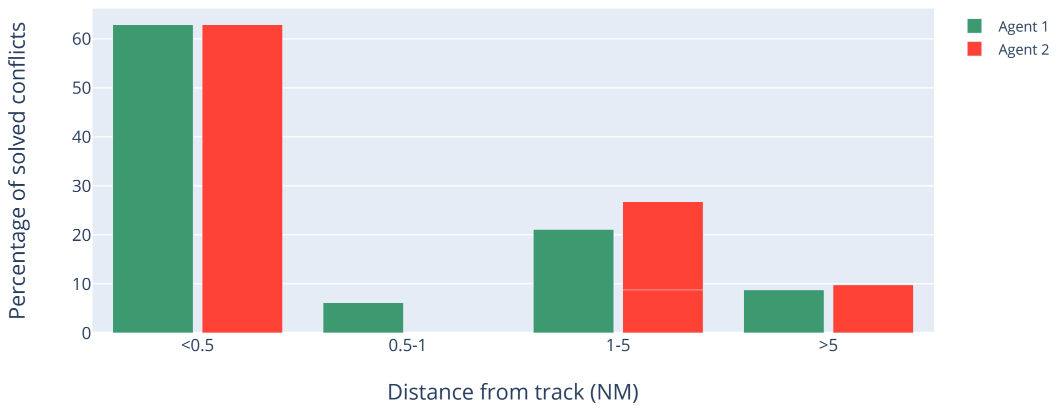


Figure 5.5: Distance of each agent from track in nautical miles when the conflict is resolved.

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 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 new scenarios were handcrafted and are not part of the original 195 test scenarios. Those were initialized prior to LOSS and the agents practically had to avoid LOSS. In the other 15 scenarios, the agents essentially had to get out of a LOSS. These results show that when given informative rewards from the environment, the agents are generally successful in solving the conflicts.

Figure 5.5 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 cannot

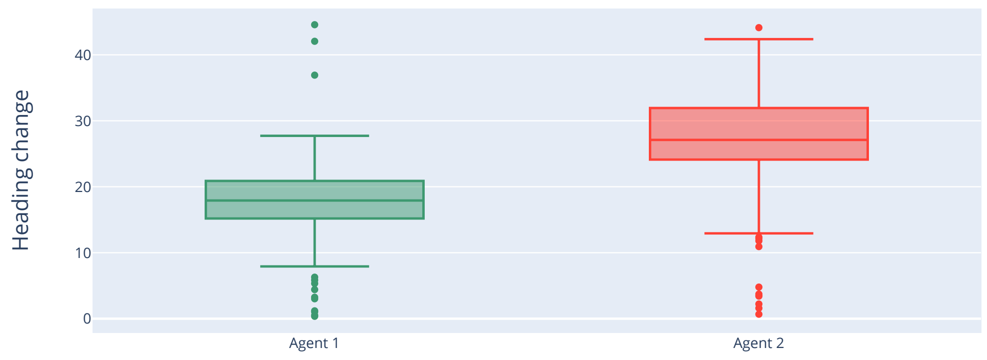


Figure 5.6: Average heading change needed to solve each conflict.

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 cannot be accepted in practice, as it would require a huge deviation, which will result in major delays.

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 5.6 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 5.7 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 [75] 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

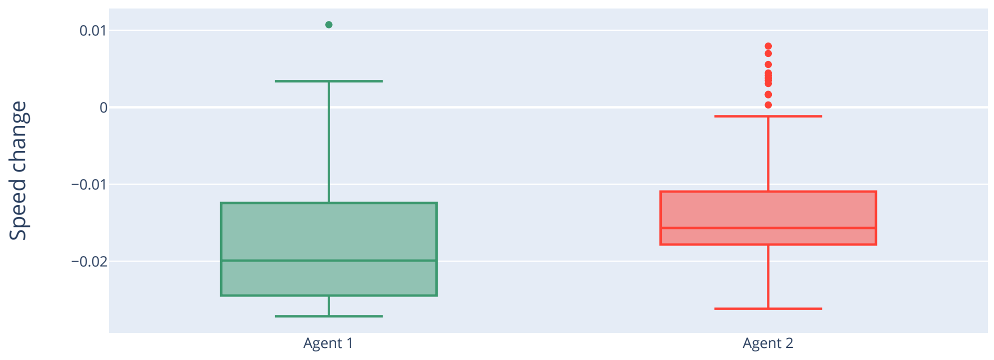


Figure 5.7: Average speed change needed to solve each conflict.

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, however this would need further investigation.

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 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.

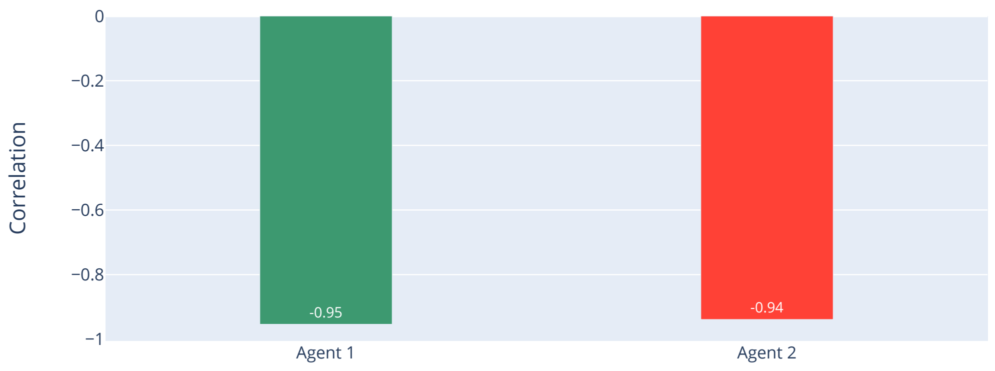


Figure 5.8: Correlation between final reward and time steps needed to solve the conflict for each agent.

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 5.6 and 5.7, 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 5.8 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

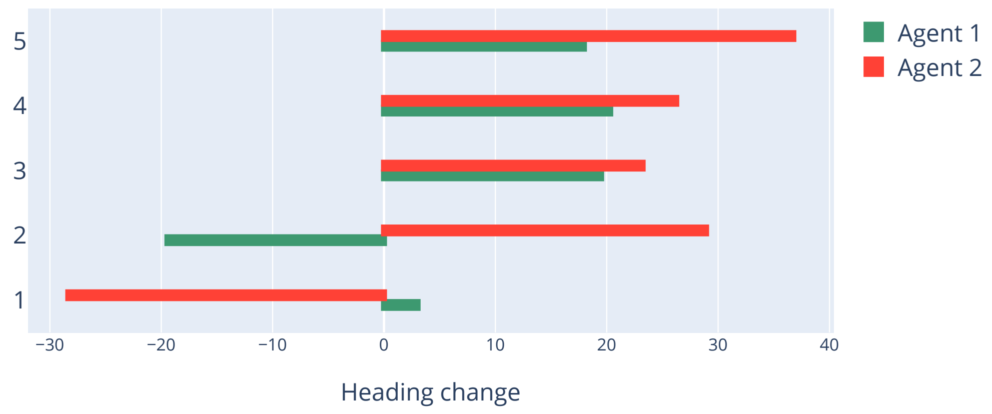


Figure 5.9: Direction and magnitude of heading of each agent for 5 example scenarios.

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 5.9 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 5.6. 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 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.

5.5 Conclusions

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 an 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. 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 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 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.

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 sent back on their tracks with the angle opposite heading and speed change used to solve 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.

Chapter 6

MULTI-UAV CONFLICT RESOLUTION WITH GRAPH CONVOLUTIONAL REINFORCEMENT LEARNING

Safety is the primary concern when it comes to air traffic. In-flight safety between Unmanned Aircraft Vehicles (UAVs) is ensured through pairwise separation minima, utilizing conflict detection and resolution methods. Existing methods mainly deal with pairwise conflicts, however, due to an expected increase in traffic density, encounters with more than two UAVs are likely to happen. In this paper, we model multi-UAV conflict resolution as a multiagent reinforcement learning problem. We implement an algorithm based on graph neural networks where cooperative agents can communicate to jointly generate resolution maneuvers. The model is evaluated in scenarios with 3 and 4 present agents. Results show that agents are able to successfully solve the multi-UAV conflicts through a cooperative strategy.

6.1 Introduction

Commercial and civil unmanned aircraft systems (UAS) applications are projected to have significant growth in the global market. According to SESAR, the European drone market will exceed 10 billion annually by 2035, and over 15 billion annually by 2050 [137]. Furthermore, considering the characteristics of the missions and application fields, it is expected that most market value will be in operations of small UAS (sUAS) and at the very-low-level airspace (VLL). Such a growing trend will be accompanied by an increase in traffic density and new challenges related to safety, reliability, and efficiency. Therefore, the development and implementation of conflict management systems are considered preconditions of integrating UAS in the civil airspace. Most notably, the National Aeronautics and Space Administration (NASA) in the USA aims to create a UAS Traffic Management (UTM) system that will make it possible for many UAS to fly at low altitudes along with other airspace users [138]. Europe is leading efforts to develop an equivalent UTM concept, referred to as U-space. It will provide a set of services (and micro-services) that would accommodate current and future traffic (mainly but not limited to) at VLL airspace [139]. Similar approaches are followed also in China and Japan [140].

Considering airspace under UTM services, UAS must be capable of avoiding static conflicts such as buildings, terrain, and no-fly zones and dynamic conflicts such as manned or unmanned aircraft. Here, a pairwise conflict is defined as a violation of the en-route separation minima between two UAVs [141]. To ensure operations free of conflict, UTM provides Conflict Detection and Resolution services, which comprise three layers of safety depending on the time-horizon (i.e., look-ahead time) [142]: Strategic and Tactical Conflict Mitigation and Collision Avoidance (CA) [141, 142]. In this work, we will focus on tactical CR applicable for small UAS missions. This function is typically treated in two ways: self-separation and Collision Avoidance [142, 143]. The former is a maneuver executed seconds before the loss of separation minima, characterized by a slight deviation from the initial flight plan, and aims to prevent CA activation. The latter provides a last-resort safety layer characterized by imminent and sharp escape maneuvers. Both functions above are encompassed within what is widely recognized as Detect and Avoid capability [144, 145]. Aligning with the up-to-date state-of-the-art, a loss of separation minima is referred to as Loss of Well Clear (LoWC). While there is no standard definition of Well Clear (WC), two related functions are associated with this state: Remain Well Clear (RWC), and Collision Avoidance (CA) [146]. In terms of tactical CD&R, RWC is equivalent to the self-separation function. Defining and computing RWC thresholds are an open research works, but they are mainly viewed as protection volume around UAS [147, 148, 149]. This volume can be specified by spatial thresholds, temporal thresholds, or both at the same time. We follow the hockey-puck model [150, 151] characterized by distance-based thresholds. In addition, the near-mid-air-collision (NMAC) represents the last safety volume. As the name suggests, a distance smaller than NMAC represents a very severe loss of well clear that could result in a collision in the worst case. This distance is usually defined based on the dimensions of the UAS and its navigation performance [152].

There are many existing works that propose conflict resolution algorithms (see Section 6.2 for a more detailed overview). However, the majority of these works focus mainly on pairwise conflicts. Nevertheless, with the expected increase in traffic density [137] multi-UAV conflicts (i.e., involving more than 2 UAVs) are expected to occur. In this paper, multi-UAV conflict resolution is modeled as a multiagent reinforcement learning problem (MARL). More specifically, we utilize graph convolutional reinforcement learning [153], where air traffic is modeled as a graph. The present UAV are the set of nodes, and single pairwise conflicts form the set of edges in the graph. The model used in this paper provides a communication mechanism between connected nodes in the graph. Such a mechanism facilitates learning and allows for the agents (in this work, an agent is an abstraction of a UAV. An agent must learn how to achieve a policy that solves conflicts through training in several multi-UAV conflict scenarios. The generated maneuvers are then forwarded to the UAVs in the actual scenario) to develop cooperative strategies. Multi-UAV conflicts are formally defined as compound conflicts, where multiple pairwise conflicts have tight spatial and temporal boundaries. In this work, we pose two research questions:

- **RQ1** Can multi-UAV conflicts be solved by modeling conflict resolution as a MARL problem?
- **RQ2** Do the agents learn any strategies in the conflict resolution process?

In this work, we first train a model in scenarios with three UAVs. After that, the same model is retrained to solve compound conflicts with four UAVs. This technique allows us to reuse the previously learned policies and refine them to a new set of scenarios, while efficiently training the new agent from scratch. Results show that agents are successfully able to solve compound conflicts in both cases.

The rest of the paper is organized as follows: some existing works are discussed in Section 6.2. Section 6.3 describes the theoretical background necessary for this paper. In Section 6.4, the experimental setup is presented. Results are presented and discussed in Section 6.5, while in Section 6.6, we draw conclusions and propose steps for further research.

6.2 Related Work

There are many essential contributions in the area of conflict resolution methods in aviation. These methods are widely classified into the geometric, force field methods, optimized trajectory, and Markov Decision Process (MDP) approaches (probabilistic) [154]. For detailed and comprehensive information on CD&R practices, we suggest Kuchar and Yang's review study [10] and this review paper [155] for more up-to-date content. We will focus only on the MDP method and provide a summary discussion below, as our work aligns with this group of methods. Aircraft and especially UAS operations are characterized by uncertain environments and stochastic events such as weather, multiple intruders, and Communication, Navigation, and Surveillance (CNS) failures; therefore, decision-making methods that adapt under such conditions are necessary. MDP and more recent Partial Observable MDP

(POMDP) are methods that can have significant performance in such domains. Different techniques are used to solve MDP and/or POMDP problems, and most notable are reinforcement learning (RL) and deep reinforcement learning (DRL) methods. In Ref. [156], the authors present an efficient MDP-based algorithm that provides self-separation functions for UAS in free airspace. A similar approach is followed here [157], where the authors give a scalable multiagent computational guidance for separation assurance in Urban Air Mobility. In addition, they use RL techniques to solve the MDP problems. In a previous work [158], a conflict resolution system is applied to mitigate conflicts between UAS. Ribeiro et al. [121] consider a single-agent approach to conflict resolution through RL for unmanned aerial vehicles (UAVs). Furthermore, recent works saw the engagement of DRL methods, which behave better in multiagent environments and consider uncertainties. In this paper [91], the authors model pairwise conflict resolutions as a multiagent reinforcement learning (MARL) problem. They use Multiagent Deep Deterministic Policy Gradient (MADDPG) [122] to train two agents, representing each aircraft in a conflict pair, capable of efficiently solving conflicts in the presence of surrounding traffic by considering heading and speed changes. In Ref. [57], the authors use the Deep Deterministic Policy Gradient (DDPG) technique to mitigate conflicts in high density scenarios and uncertainties. Brittain et al. [55] used a deep multiagent reinforcement learning framework to ensure autonomous separation between aircraft. Dalmau et al. [62] used Message Passing Neural Networks (MPNN) to model air traffic control as a multiagent reinforcement learning system where agents must ensure conflict free flight through a sector.

While these papers consider a multiaircraft (manned or unmanned) setting, they do not particularly consider small UAS performance capabilities (i.e., high yaw rate). Also, a common assumption is that the flight trajectories should be within a predefined airspace sector. In a UTM environment airspace is not necessarily segregated into sectors. Additionally, small UAS characteristics can directly effect how the action space is modelled. Moreover, approaches with a multi-UAV setting do not consider the effects of cooperation on the resolution manoeuvres. In this work, we propose a multi-UAV conflict resolution method suitable for sUAS operations and attempt to achieve cooperation between the agents.

These methods (RL and DRL) are considered very important for the development of the Aircraft Collision Avoidance System (ACAS-X), which will be extended into ACAS-Xu, ACAS-sXu, and so on, to accommodate all airspace users [159].

6.3 Theoretical Background

In this section, we will briefly describe the main theoretical background necessary for our work. However, it is not feasible to properly cover all necessary details. Therefore, we refer the readers to some works that give a comprehensive explanation of the concepts used in our work [123, 160, 161].

Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of machine learning which deals with sequential decision making [123]. A given RL problem is formalized by a Markov Decision Process (MDP), which is a discrete time stochastic control process [125] 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 probabilities between states,
- $R : S \times A \times S \rightarrow \mathbb{R}$ 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:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{6.1}$$

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 behavior 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 [124].

The RL agents 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 states then transitions to $s_{t+1} \in S$, 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 behavior is encoded into a policy π, which can be deterministic $\pi : S \longrightarrow A$, or stochastic $\pi : S \times A \longrightarrow [0, 1]$.

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

$$V^{\pi}(s) = \mathbb{E}_{\pi}(R_t | s_t = s) \tag{6.2}$$

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}(R_t | s_t = s, a_t = a) \tag{6.3}$$

The value function represents the future expected reward in the current state if the policy π 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 [62]. 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 [125].

In the case of Deep Q-Networks (DQN) [118], the Q function approximators are neural networks. However, several issue 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. Experience replay is used to mitigate the issue of sample autocorrelation [118]. 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 [126], which are decision processes for $\mathbb{N}$ 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. 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.

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 [162]. 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) [163], Graph Convolutional Networks (GCNs) [164] and Message Passing Neural Networks (MPNNs) [165]. We refer the reader to [162], for a comprehensive review of GNNs.

In the case of MARL, communication is often cited as a key ability for cooperative agents [153, 62]. In such a setting, agents exchange information before taking an action.

In this work, we will use Graph Convolutional Reinforcement Learning [153] (dubbed DGN by its authors), which is a GNN algorithm for cooperative agents.

In DGN, the multiagent environment is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ 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 neighbors $\mathbb{B}_i$, where $(i, j) \in \mathcal{E}, \forall j \in \mathbb{B}_i$. The set of neighbors 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 [166], 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 [153, 166] for a detailed overview of the attention mechanism). Let us denote with $\mathbb{B}_{+i}$ the set of neighbors $\mathbb{B}_i$ 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 $i, j \in \mathbb{B}_{+i}$ is as follows:

$$\alpha_{ij}^m = \frac{exp(\tau \cdot W_Q^m h_i \cdot (W_K^m h_j)^T)}{\sum_{a \in \mathbb{B}_{+i}} exp(\tau \cdot W_Q^m h_i \cdot (W_K^m h_a)^T)))} \tag{6.4}$$

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:

$$h'_i = \sigma(concatenate(\sum_{a \in \mathbb{B}_{+i}} \alpha_{ij}^m W_V^m h_a, \forall m \in M)) \tag{6.5}$$

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 $\mathbb{B}_i$ and 0 for any agents not in the neighborhood of i. The feature vectors are merged into a feature matrix F with size $N \times L$ 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 $C_i \times F$.

The Q network in DGN is a common network as described in II.B. 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 isthe 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:

$$\mathcal{L}(\theta) = \frac{1}{S}\sum_{S}\frac{1}{N}\sum_{i=1}^{N}(y_i - Q(O_{i,C,a_i};\theta))^2 \tag{6.6}$$

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:

$$\theta' = \beta\theta + (1 - \beta)\theta' \tag{6.7}$$

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

6.4 Experimental Setup

Compound Conflicts

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. [130], 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 [130].

Traffic as a Graph

In this work, the multiagent environment is represented as a graph. Therefore, we must define how the graph is created for a given traffic scenario. To have a correct definition of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the set of nodes and edges is required.

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 neighbors first and foremost. Therefore, we make this choice to facilitate cooperation between UAVs that are in conflict.

Training Environment

To model compound conflict resolution as a MARL problem, the underlying Markov decision process must be formalized. Thus, we have to determine the state space, action space, and reward function. As we are considering cooperative agents, the ultimate goal is to maximize the joined reward. Therefore, all agents have the same reward structure.

State Space

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. [91], 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.

Action Space

In this work, we only consider solutions through heading changes, thus speed and altitude changes are ignored. As such, agents can choose to take on 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° in either direction. Agents must make a decision every 2 s.

Reward Function

Once the agents take an action according to their policy, they will receive a reward from the environment $r_{i,t}$ for the current time step, which indicates the quality of the action. Thus, a carefully constructed reward function is crucial in achieving desirable performance [91].

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° 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:

$$
\begin{aligned}
r_{i,t} = w_1 \sum_{\mathcal{E}(i)} -1 + w_2 &\begin{cases} -\frac{|\mu-\mu'|}{90} & \text{if } |\mu-\mu'| < 90 \\ -10 & \text{otherwise} \end{cases} \\
&+(-w_3(1 - exp(1 - \frac{1}{(\frac{d_{cpa}}{d_{thresh}})^{1/2}})))
\end{aligned}
\tag{6.8}
$$

where w_1, w_2, w_3 are positive weights that indicate the importance of each term, $\mathcal{E}(i)$ indicates all the agents that have an edge with i, μ and μ' are the original and current heading and d_{cpa} and d_{thresh} are the distance at CPA self-separation distance. The formulation of the *severity term* is used to exponentially penalize conflicts with higher severities. It ranges from 0 to 1. In this paper, w_1, w_2, w_3 are kept equal, however in future work these can be extended to be learnable parameters. The total reward for a given time step t is:

$$
R_t = \sum_{i}^{N} r_{i,t} \tag{6.9}
$$

where N is the total number of agents.

Simulation Environment

Simulations were run on the Air Traffic Simulator BlueSky [109]. The simulator was chosen primarily because it is an open-source tool, allowing for more transparency in developing and evaluating the proposed model. Furthermore, BlueSky has an Airborne Separation Assurance System (ASAS), supporting different CD&R methods. This allows for different resolution algorithms to be evaluated under the same conditions and scenarios.

Data Generation

Algorithm 4 describes the procedure to generate the training scenarios. In this work, we consider compound conflicts with 3 and 4 UAVs. 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. To define the metrics for self-separation, we follow a similar approach as in [167, 168]. This threshold depends on the UAV maneuverability and its maximum airspeed. Whereas the innermost layer will be modeled according to the Near Mid Air Collision concept, as a circle with radius: $R_{NMAC} = 2 \times$ Maximum Wing Span + Total System Error (TSE). The self-separation can be calculated by (10).

Algorithm 4 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
```

$$R_t = R_{NMAC} + V_m \times t_m + \frac{V_m}{\omega_m} \tag{6.10}$$

where V_m and ω_m are maximum airspeed and maximum yaw rate, respectively, while t_m is the time needed for the UAV to make an avoidance maneuver. The self-separation threshold was set to 240 m, taking into account R_{NMAC} = 4 m, maximum airspeed 15 m/s, and a maximum yaw rate of 90°/s As we are attempting to solve conflicts at the tactical level, a duration of 1 min per scenario was deemed suitable. The time metrics (i.e., tactical CD&R maneuver and look-ahead time) mentioned above are synthesized from the state of the art of CD&R in small UAS [169, 148, 170].

6.5 Simulation Results

Conflict Resolution Performance

The model (the code can be found at `https://github.com/risufaj/bluesky`, Accessed on 08/01/2022) 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 on the Google Cloud Platform (`https://cloud.google.com`, Accessed on 08/01/2022) using an NVIDIA Tesla K80 GPU. The training lasted around 10 h.

Figure 6.1 shows the evolution of the cumulative reward for both cases. As the agents are cooperative, we are interested in the overall reward that is gained per episode and do not concern ourselves with the individual rewards. In this work, we utilize negative rewards, so the maximum the agents can get is 0. In the case of the 4 agents the reward seems a bit lower, however this comes a result of there being one more agent present, which takes actions to solve the conflicts thus inflicting itself some negative rewards for going away from track.

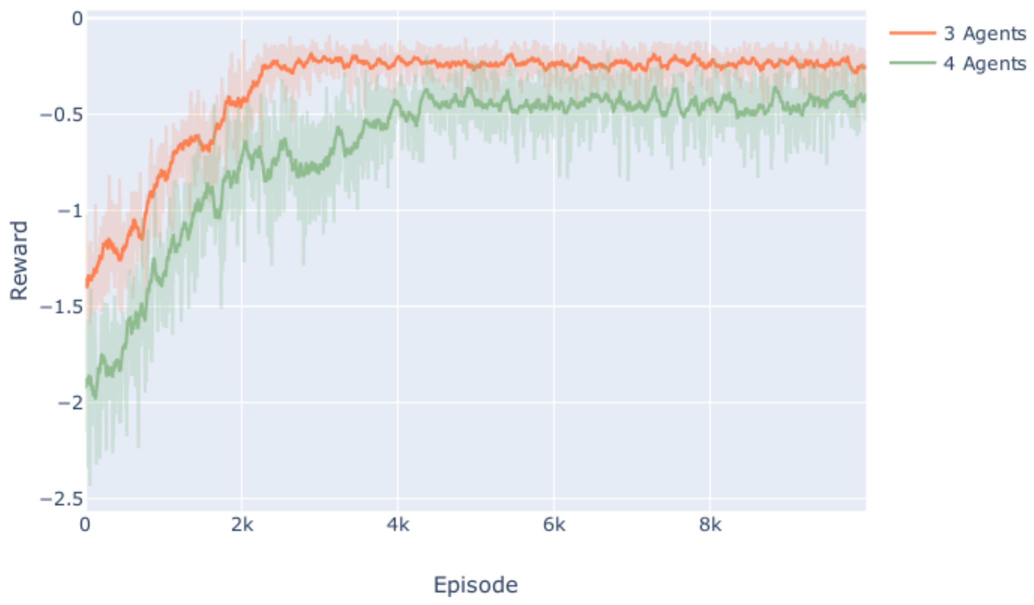


Figure 6.1: Evolution of cumulative reward per episode.

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 with 3 present agents, the convergence happens around 2000 episodes, while around 4000 episodes are required for the 4 agent case. In the latter case, 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.

In Figure 6.2 the number of losses of separation (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 LOSS. In the case with 3 present 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 the 4 agents compound conflict, the average is around 1 LOSS per episode. However, through our results, we note that the model manages to always avoid near misses in both cases, as the NMAC distance is never breached.

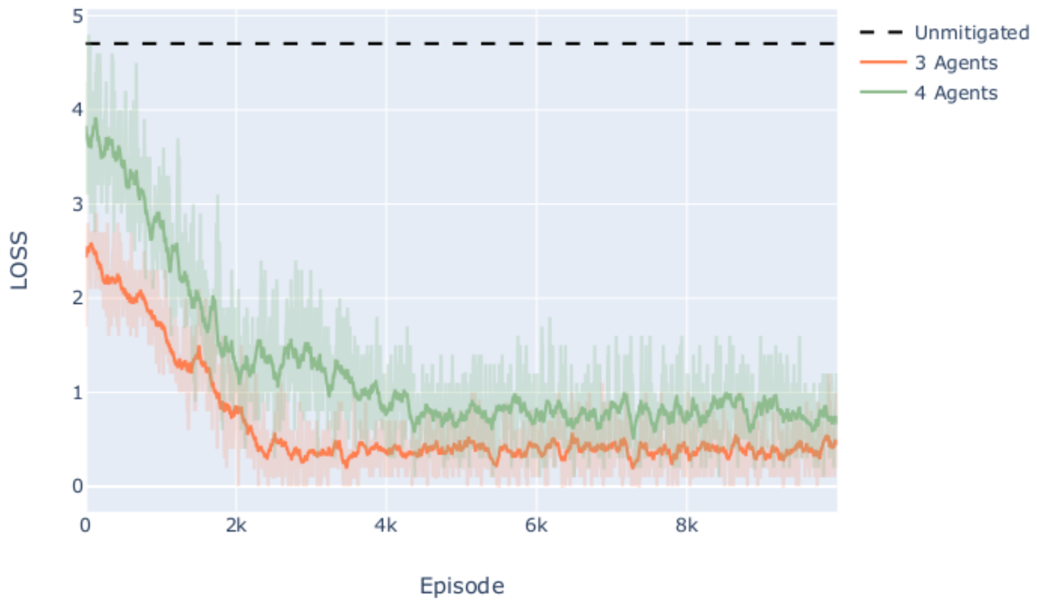


Figure 6.2: Number of losses of separation in comparison with average unmitigated case.

We can observe a similar evolution in Figure 6.1, with the retrained model performing slightly worse. In general, the case with four agents had more pairwise conflicts present, which makes the problem more difficult.

Table 6.1 shows in how many episodes the compound conflict was solved, meaning no LOSS has occurred. The difference is similar to the number of extra epochs the 4 agents model needed to converge. As such, once the model converges it can generally manage to solve the compound conflict, thus fulfilling its task successfully. This result shows that the agents are able to solve conflicts through communicating with their neighbors.

Table 6.1: Number of episodes compound conflicts solved.

3 Agents	**4 Agents**
5650	4461

In addition to solving conflicts, it is desirable for agents not to spend too much in a LOSS, as this can increase the risk of collisions. Such information is shown in Figure 6.3. The results shown there further confirm that the agents are able to improve their performance. In a similar trend, the case of the 3 agents compound conflict seems simpler to solve successfully, as the agents spend less than 5 s in a LOSS, with 5650 episodes not experiencing a LOSS (therefore no time steps in LOSS) Through the results presented in this section, **RQ1** can be answered. In both problem settings, agents can improve and eventually solve the majority of conflicts. Furthermore, even in cases where there are still conflicts present at the end of an episode, we observe that there are no NMACs present. Nevertheless, an approach where hard-coded maneuvers (i.e., Collision Avoidance) as a second layer of safety can be included.

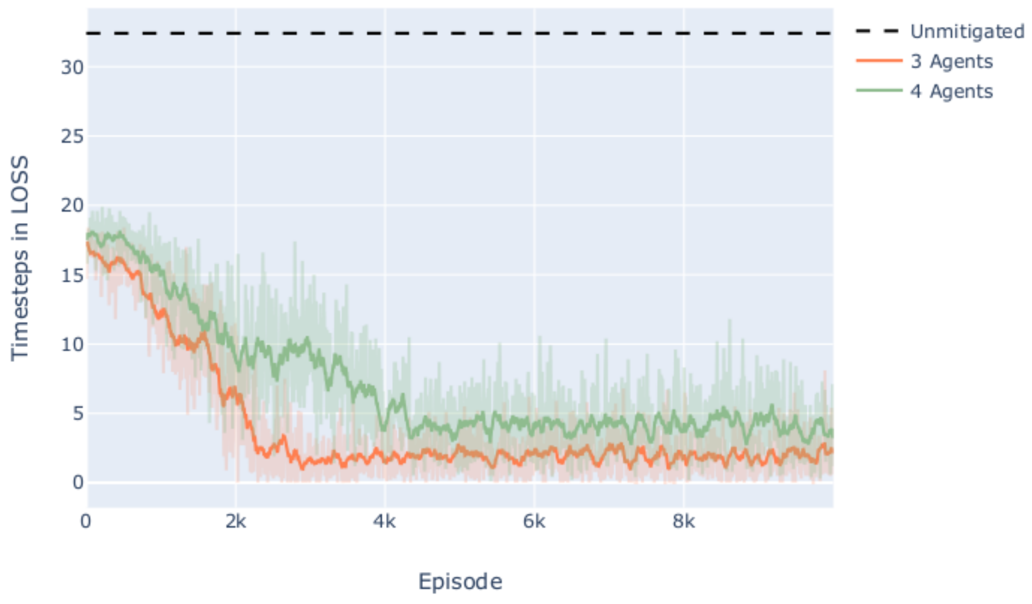


Figure 6.3: Number of time steps spent in LOSS.

Agent Behavior

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 6.4 and 6.5. 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.

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. These results answer **RQ2**.

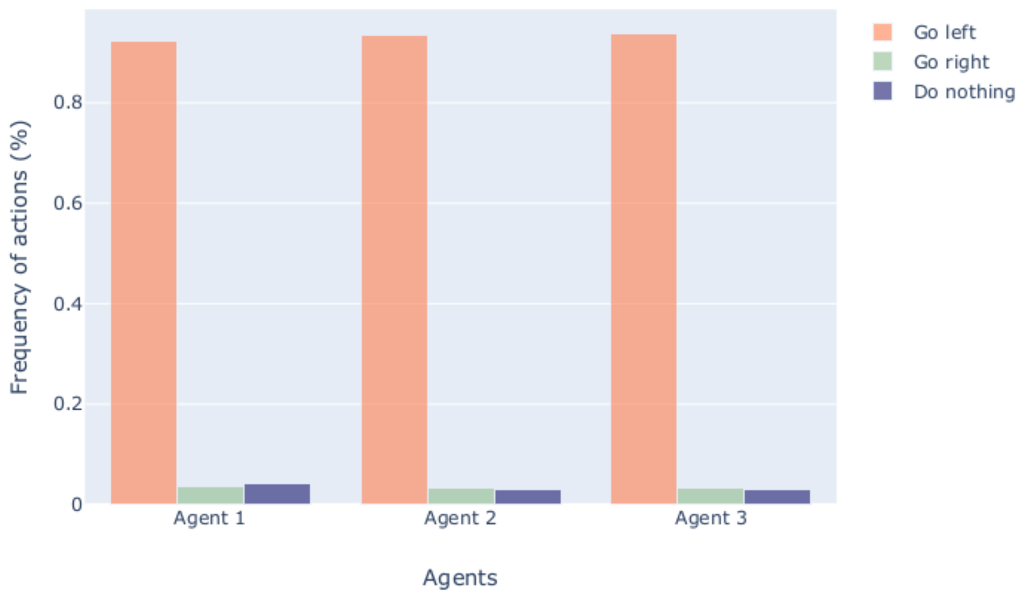


Figure 6.4: Frequency of actions for last 200 episodes of compound conflict with 3 agents.

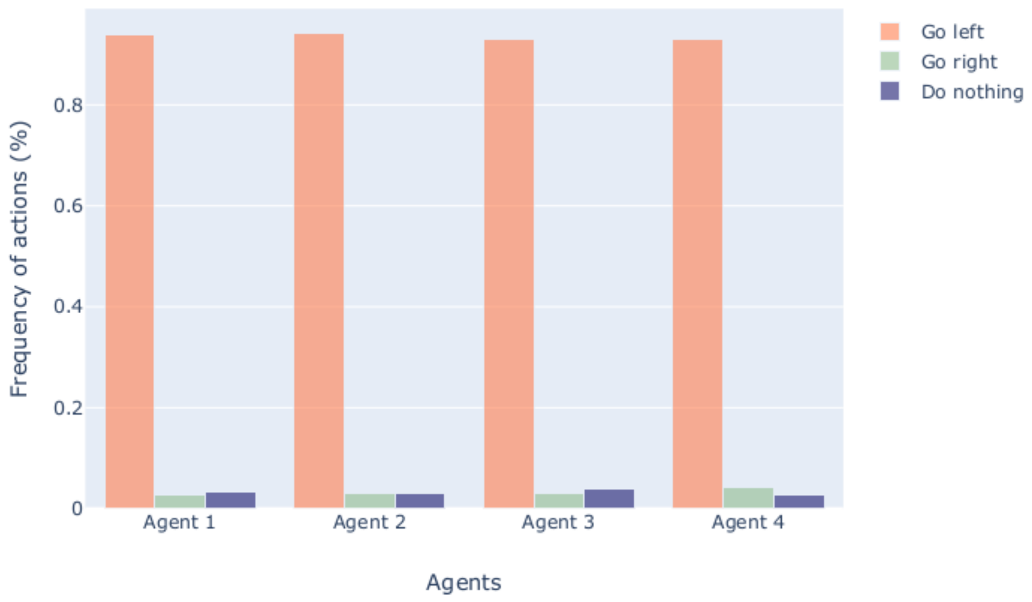


Figure 6.5: Frequency of actions for last 200 episodes of compound conflict with 4 agents.

6.6 Conclusions and Future Work

In this paper, we tackle multi-UAV conflict resolution by modeling it as a MARL problem with cooperative 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.

We first train a model that learns how to solve compound conflicts with 3 agents. After that, the same model is retrained to to solve compound conflicts with 4 agents. As a result, we are able to refine the policies learned in the previous setting, while added agent learns a desirable policy.

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 $\mathcal{N}$ 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.

Chapter 7

CONCLUSIONS

7.1 Contributions

The contributions made in this thesis can be grouped in two topics: air traffic complexity and Multi-Agent Reinforcement Learning (MARL) for conflict resolution. For each topic, several key topics were identified which were not covered by existing research. This section serves to summarize all individual contributions.

Air Traffic Complexity through Graph Theory

Contribution: Air traffic complexity has been defined in terms of graph connectivity

First of all, air traffic has been modelled as a dynamic graph. The present aircraft in a given airspace volume in time t are the set of vertices for the graph at that time, while the interdependencies between the aircraft are the set of the edges. We formalized four indicators based on graph theory to measure the complexity of air traffic. These indicators combine topological information gathered from the geometries of interdependencies, with the severity of interdependencies to present a full and nuanced picture of complexity. Furthermore, we consider the dynamic evolution of complexity and do not simply give one single complexity score. The achieved results (see Chapter 2 for a complete overview of the results) show that the proposed indicators give detailed information and thus overcome drawbacks of existing metrics. Furthermore, these indicators were also applied to high density scenarios for sUAS (see Appendix A)

Contribution: The concept of single aircraft complexity

An important feature of three of the four indicators is that they can be measured for every aircraft separately. Thus, our previous work was further developed by extending the existing indicators to introduce the concept of single aircraft complexity. Given a certain airspace, we define the complexity of an aircraft as the average contribution of this aircraft to each indicator. Therefore, we are able to calculate the percentage that each aircraft contributes to the overall complexity of the airspace. As with all the indicators, this percentage can be computed along time, providing a dynamic view of the complexity of each aircraft. Such a definition allows for a comparative analysis of which aircraft present at a particular time is the most complex, as well as an evolution of this complexity.

Contribution: Defining complex communities using single aircraft complexity

Furthermore, using this concept, we are able to identify complex spatiotemporal areas in a sector by defining complex communities. A community is defined as a connected component of a graph, while a complex community must be responsible for more than a certain threshold of the overall complexity. The complexity contribution of a community is the sum of the individual contributions of the aircraft that are members of the community. In order to deal with the evolution of these communities, we introduce an algorithm that is able to persist communities in time by using the Jaccard similarity in consecutive time steps between communities. Some parameters, such as the threshold when interdependencies start forming or when a community is complex are modelled to be specific to the problem or even the user. As such, different ATM actors can set different thresholds depending on the problem they are tackling using the proposed methodology. An extensive sensitivity analysis was conducted to quantify how each input affects the different outputs of the methodology.

Contribution: Developed a web application as a tool to showcase the developed methodology

To showcase how the methodology developed in this work can be used, a web application was developed. This web application visualizes different outputs of the methodology and a summary containing a statistical information can be downloaded. Through the different use cases, We first show that the algorithm can serve to formalize controller decisions as well as guide controllers to better decisions in situations where multiple pairwise conflicts co-exist in time. Further, we investigate how the provided information can be used to increase transparency of the decision makers towards different airspace users, which serves also to increase fairness and equity.

Conclusion: Our work on air traffic complexity shows through use cases and empirical results the effectiveness of modelling air traffic as a graph. The use of graph theory as proposed in this work can give a full and nuanced view of complexity while inherently allowing for the definition of desirable characteristics such as the subjectivity of complexity.

Conflict Resolution as MARL

While machine learning methods have been used for conflict resolution, we improve on the application of these methods. Air traffic is a highly dynamic environment with different actors participating in it. These actors have different interests. In our work, this is modelled as a Multi-Agent Reinforcement Learning (MARL) problem. In such a setting, which is an extension of single agent RL, several agents interact with an environment and amongst themselves in order to maximize some notion of cumulative reward. The behaviour of each agent is encoded in a policy, which maps states to actions. Initially, applied MARL for the purpose of tackling conflict resolution in the context of en-route traffic at the tactical level. In our approach we considered pairwise conflicts together with surrounding traffic.

Contribution: State representation based on the properties of deep learning

In order to model conflict resolution as a MARL problem, we have to formalize the underlying MDP. In order to do so, we have to model an environment, define the action and state space for each agent as well as the reward function. In this work, we make no assumptions about the discretization of space, therefore we consider a continuous state space. Consequently, it is impossible to use traditional RL methods, where the reward is stored for each visited state. In our work, this requires storing rewards for a potentially infine number of states. Therefore, it is necessary to turn to Deep Reinforcement Learning (DRL), where the policy is represented by neural networks, which attempt to approximate the optimal policy. Such a decision allows us to avoid the lengthy process of feature engineering. Thus, we propose a novel state representation consisting of position information, heading and current speed.

Contribution: Reward function that considers multiple factors and the interests of different ATM actors

We propose a reward function that not only optimizes for number of conflicts solved, but encourages efficient solutions that consider the preferences of different actors. The reward function contains factors such as fuel consumption, CPA, time to LOSS etc. As a result, our work in this thesis should also be seen as a baseline for other research that goes in the same direction.

Contribution: Continuous action space

Another important aspect in formalizing the underlying MDP is the modelling of the action space. Related works often consider a discrete action space and only solve conflicts using either heading changes, speed changes or altitude changes. In our work, we propose a model that can handle continuous action spaces and we enable solutions to be possible via speed and heading changes combined. We avoid altitude changes for two reasons: first, they are usually not preferred by ATCos and to decrease the computational burden to train our model.

Contribution: Conflict resolution using MARL with real historical traffic

The model was trained and tested using real historical traffic taken from Eurocontrol's DDR II dataset. In order to increase the variance of the data, a data augmentation technique was developed (detailed in Chapter 4). The achieved results have been very positive (see Chapter 4 for a complete overview of the results). Most importantly we outline that the model could solve the vast majority of conflicts while showing desirable behavior such as solving the conflicts quickly and preferring heading changes.

Nevertheless, several limitations were identified. First of all, with the increase of traffic and the level of automation in ATM, it is desirable that ML handle also situations with more than 2 agents. This proved to be very difficult using the previously used model as introducing new agents resulted in the model not being able to converge. Furthermore, we identified several cases where the agents behaved in peculiar ways; for instance, if agents were unable to resolve the conflict quickly it was very likely the conflict could turn into a loss of separation. Finally, further investigation showed that while continuous action spaces provide more freedom in what action agents can take, it comes with increased computational needs, resulting in longer training times.

Contribution: Graph Neural Networks to overcome limitations of early MARL model

Contribution: Scalable conflict resolution

We attempted to overcome these shortcomings by utilizing GNNs. They are a recent advance in DL which have proved very effective in graph data. To employ them in a MARL setting, we use Graph Convolutional Reinforcement Learning. In this work, we tackle multi-UAV conflict resolution with cooperative agents in situations with 3 and 4 present agents. Similar to our other work, air traffic is represented as a graph with aircraft as nodes. Edges are created between aircraft of a pairwise conflict. By using GNNs, which provide a communication mechanism between connected agents, the development of cooperative strategies is facilitated. To formally define a multi-aircraft conflict, we propose the concept of compound conflicts, which are conflicts that have spatial and temporal interdependencies.

We maintain several modelling choices as in our previous work (e.g., air traffic as a graph, using DRL, multi-factor reward function), while overcoming the shortcomings of our previous work (see Chapter 5 for a complete overview of the results). In our view, the most important result of this particular work is in showing that we are able to scale up MARL methods while enabling the agents to learn meaningful cooperative strategies. More specifically, agents effectively learn to solve the more severe conflicts first and then solve the remaining conflicts while making sure that no new conflicts are created. Such a strategy is replicated when more agents are added in the environment.

Conclusion: Through the work in this thesis we show the effectiveness of using MARL for conflict resolution. Several contributions in terms of designing complex reward functions, continuous action spaces and scalability through the use of GNNs have been made. This work opens the door to the application of this paradigm of AI in aviation.

7.2 Future Work

Given the breadth of the research conducted during this thesis, there are several different directions through which our work can be extended. We will list some of these directions and outline the work required.

Different sources of uncertainty

Throughout the work of this thesis, no sources of uncertainty in the positions of the aircraft are considered. However, in realistic traffic, there are several factors that can affect the trajectory or the velocity of aircraft. For instance, wind, uncertainty from sensors, communication delays and so on are factors that should not be ignored in real practical scenarios. There are several ways our work can account for such uncertainties. First of all, these uncertainties can be modelled as probability distributions. When considering the effects of uncertainties in air traffic complexity, we propose Monte Carlo simulations as a way to approximate the expected value of each indicator. The concept behind Monte Carlo simulations is to sample some distribution to solve problems that might be deterministic in principle. If the number of simulations is high enough, we can provide a distribution of each complexity indicator over time which takes into account different sources of uncertainty in the input.

On the other hand, such uncertainties could be used by the MARL models to provide valuable information about the robustness of these models. For instance, when uncertainties are considered, an action can take an agent to a different position than originally planned. Thus, agents must be capable of making decisions that account for this uncertainty and are still able to maintain safety.

Considering self-interested and heterogeneous agents

In this thesis, we have only considered MARL models for conflict resolution. Through the achieved results, it has been shown that these models are in fact capable of solving difficult conflicts that involve multiple aircraft. Therefore, we believe that these methods should be further investigated in bigger time windows. Such a change in setting has important implications for the goals and behaviors of the trained agents. For instance, in our work, the agents have been working with a single common goal: safety. This is an acceptable assumption given the short window that we have been considering. However, when extending this window, it will be necessary to consider not just a common goal, but the individual goals of agents.

Let us take the scenario of dynamic re-routing in the case of UAS. In such a scenario, the goal is to modify the trajectory of some UAS in order to optimize some performance indicator: be that safety, capacity, complexity etc. If this problem is modelled as MARL, individual agents must be able to reach their destinations while considering common goals. Furthermore, there could also be some restricted areas which the agents cannot access. On the other hand, such a scenario might violate another common assumption of our work: cooperativity. While in the case of one common goal, this is a completely reasonable assumption, in a scenario with longer time windows and more complex goal definitions, this assumption might not hold.

There are several differences in problem setting that can have a big impact on the underlying MARL problem. A possible difference could be that not all agents are willing to communicate or are limited in the information they share (or wrong information in the case of malicious agents). Several issues arise in this case. First of all, we must account for the fact that the agents in the environment are no longer homogeneous. A possible way to tackle this issue would be to consider hereogeneous graphs[1], where there can be different types of vertices and edges. There is already work in applying GNNs to these types of graphs, but not in the

[1]We believe that GNNs and air traffic as a graph in general, are important tools that should further investigated in the field

context of RL. Therefore, we believe it could be an important direction to follow. In the same vein, keeping the same example of dynamic re-routing, agents can choose when they cooperate and with which agents they cooperate. Let us consider a scenario where aircraft of the same operator can communicate with other fellow aircraft of that operator. In such a scenario, which can be driven by the competition between operators, Thus, they can view the reward between them and other aircraft as a zero-sum game, resulting in competitive behaviour between agents. Such a behaviour is rarely investigated in literature, but can naturally arise in the context of aviation.

On the other hand, agents might not be able to perform the same actions, for example in the case of different types of aircraft in the airspace. In such a case, it is not enough to model the different types of edges between the aircraft, the underlying MDP must be modelled differently. While there are several works that consider such cases, to the best of our knowledge, there are no previous works that combine this with GNNs.

While we only consider the example of dynamic re-routing, we believe that particular characteristics of any given problem in aviation can result in important questions also for MARL. Thus, to summarize, it is necessary to investigate what strategies self-interested, heterogeneous and competitive agents can learn.

Air traffic complexity as a basis for designing better CR algorithms

The indicators provide a new framework in the design of conflict resolution algorithms in order to preserve safety while reducing traffic complexity. Complexity is usually not taken into account in designing such algorithms (see results of our work in Appendix A), which can lead to solutions that can make the situation difficult for the controller to handle. Additionally, by using the indicators, algorithms can be tuned to encourage resolutions that lead to lower complexity and discourage those that increase complexity. Such algorithms can be used to improve the quality of resolutions, in addition to solving the present conflicts. Furthermore, the concepts of single aircraft complexity and complex communities, could allow for a more granular optimization of complexity providing more detailed information.

Industrialisation

We believe the work performed in this thesis has the potential to be used in industry in two different contexts: airspace complexity and MARL applications. For the former, the results achieved in different stages of our work have 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 thesis, 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. Finally, the certification of AI models in daily operations is an open problem, which will need extensive and careful research.

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.

Involving humans

With aviation being safety critical, it is crucial that progress in automation or autonomy be designed with the humans that must interact with these systems in mind. This is true also for our work. While the purpose of our work has been to improve upon existing works regarding complexity and conflict resolution, moving forward one has to consider the impact of these methods. We will discuss this for each subsection of our work.

Complexity in ATM has been historically linked with the workload of controllers [72, 73, 31]. This often is a intuitive connection to make, but workload is a highly subjective concept, and formalizing the link between the two has proven to be a non-trivial task. As such, we believe it is of utmost importance to validate the proposed indicators through ATCos. Such a validation can be in the form of a survey in which controllers are given different traffic geometries and asked to assess if, in their experience, this is a complex situation or not. Evidently, it is necessary that this survey be completed by a sufficient number of ATCos in order to reach statistically meaningful results. The results of such a survey could be used to establish connections between complexity as quantified by the graph indicators and the ATCo's perceptions of complexity. Nevertheless, this presents a significant hurdle, as getting enough and relevant expert opinions can be a strenuous task.

We attempt to embrace the subjectivity of complexity, by allowing the complexity threshold for communities to be an input parameter which can be set according to the user and task. Nevertheless, this is not the complete picture. In addition to evaluating the indicators, it will be important to present the information provided by our work in a meaningful way. Thus, the tool proposed in our work should be further elaborated and research into human-computer interfaces (HMIs) could be conducted to establish the best way to present this information.

On the other hand, an AI roadmap proposed European Union Aviation Safety Agency (EASA) [171] emphasizes the importance of a human-centric approach to AI in aviation. Crucially, in order to increase trustworthiness, AI needs to be transparent, fair and non-discriminative and explainable. AI is not completely reliable as it is inherently non-deterministic. Moreover, AI systems and especially DL systems are often considered as "black boxes", meaning that it is not easy to understand how it works and why it produces certain outputs. Safety in AI has been researched from different angles such as avoiding unintentional harm [172] and ethics [173] and the need for transparency, interpretability and explainability has been often emphasized [174]. In [175] the importance of these characteristics to increase trustworthiness is outlined. There, it is argued that in order to increase trust in AI models it is crucial to understand why it makes the decisions it does make. Furthermore, it is crucial that these explanations are tailored to the human who must interact with them. Additionally, these characteristics are important in understanding the system retrospectively, e.g., to understand a wrong or even harm-causing decision and proactively e.g., to predict and prevent any future harm-causing decisions.

Finally, previous research has emphasised [20, 176] that AI models must be aligned with the preferences and values of humans, while being allowed to explore other solutions (see 7.3 for a more detailed argument). It is reasonable to expect humans to have some preferences over outcomes of certain situations. Furthermore, while preferences can be hidden or implicit, evidence could be found in the choices that the human has made. In this case, the machine would attempt to solve its task while maximizing the human preference. More specifically, the AI model would need to interact with the human by asking questions [177] or additional feedback in order to reduce its uncertainty about the human preferences. In such a scenario, it is not necessary to put a fixed purpose to the model, but the model acts in such a way that it avoids potentially important but unknown preferences. As such, we believe it will be important to further condition the MARL models proposed in this thesis to be further fine tuned by aligning them with human preferences. For instance, ATCos tend to prefer resolutions based on heading changes, rather than solutions based on speed changes. Furthermore, pilots prefer heading changes to altitude changes. A possible solution could be Preference-based Reinforcement Learning (PbRL) [176], where the reward function is learned through following more preferred trajectories[2] Trajectories are labelled according to the preference of a human demonstrator, and then RL agents must learn to make decisions that satisfy this preference. Another possible solution could in the form of Cooperative Inverse Reinforcement Learning [178] where AI agents to learn human preferences directly.

7.3 A Vision of AI in Aviation

In this section, we will attempt to summarize some important lessons learned through applying AI to (manned and unmanned) aviation. According to a roadmap laid out by SESAR JU [179], there are 5 levels of automation depending to the extent of involvement of the automated tool. Most of the work about AI in aviation (including the work of this thesis) can be classified as a level 1 or 2. In order to achieve higher levels of automation, as mentioned in the previous section, we believe a crucial aspect will be the alignment with human preferences and values [20, 171, 179], which will improve the acceptability and trustworthiness of AI in aviation. It might be worth elaborating on the reasons behind such a claim.

We briefly reiterate that DL methods allow computational models that are composed of multiple processing layers to learn representations of data with multiple level of abstractions [21]. These methods discover intricate structures in large data sets by using the backpropagation algorithm (or variants of) to indicate how the machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer [21]. For instance, the use of convolutional networks has resulted in breakthroughs in image and video processing, recurrent neural networks have been used on sequential data such as text and speech, whereas graph neural networks have made possible to use conventional DL algorithms in non-Euclidean data such as social networks. However, the majority of problems that have been solved by AI so far are characterized by clear rules, a

[2]A trajectory is any finite sequence of state-action pairs.

clearly defined objective function and usually are close ended, i.e., have definitive answers. Thus, these models work best when they are entirely certain about their objectives. Such characteristics have been described as the *standard model* of AI where one builds an optimizing machine, inputs the objective and trains it [20]. Training these models is typically an expensive operation, as it can take large amounts of time and computational resources on sometimes very specific hardware such as Graphics Processing Unit (GPU) or Tensor Processing.

Why alignment is important

As we show in our work, the standard model of AI, and more specifically, the paradigm of MARL, seems to be perfectly suited for Conflict Resolution at the tactical level. More specifically, there is a clear objective function: safety; clear rules: physical capabilities of the aircraft and operational requirements; close-ended: safety is either achieved or not. Furthermore, MARL algorithms do not strictly require knowledge about the dynamics of the environment, as the agents interact with the environment in a trial-and-error fashion. On the other hand, as previously mentioned, there can be scenarios where agents are self-interested and are not only working towards a common goal. A typical examples is competing delivery companies operating in an urban areas using sUAS. In this setting, agents can have different, private reward function and the goal of cooperation is to achieve also some common sub-goal such as safety. Therefore, agents can be heterogeneous in their preferences, actions and rewards. In the former case of only a common goal, models are usually centralized where there exists a central controller that can aggregate information from the agents in the form of joint actions, observations and rewards and can also design policies for all agents. In the latter case, there is no central controller. Agents can be connected via a time-varying communication protocol such that local information can be spread across agents. In an extreme case, there can not be any communication protocol and each agents makes its decision only based on its local observations without coordination.

However, in different scenarios where some sort of long term planning is required, things become less clear. While safety remains important, there can also be different Key Performance Areas (KPAs) such as capacity or flight-efficiency that need to be considered. This results in a vague objective function to be optimized. As Russell states in ([20]) “If the objective is wrong, we might be lucky and notice the machine’s surprisingly objectionable behavior and be able to switch it off in time”. Thus, the use of AI in different levels of ATM, where planning is required could come with dire consequences. In fact, this is a known problem in AI, outlined in various works ([180],[181]) with researchers emphasizing the importance of aligning the actions of the machine with our own goals in situations where our goals are not entirely clear. As stated, we need to put in the right goal or purpose. In ([20]) this is called the King Midas problem: Midas got exactly what he asked for, namely, that everything he touched would turn to gold but, he discovered the drawbacks of drinking liquid gold and eating solid gold. An AI tasked with planning trajectories could delay or never allow aircraft to reach their goals if it is penalized too heavily for safety.

Finally, the ethical and legal implications of an autonomous ATM system must be considered. With ATM being safety critical, the nondeterministic nature of AI is simply too big of a problem to be ignored. The simplest question to ask in this case is: What should happen if the AI is not able to avoid a collision? According to other researchers [20, 181] the most straightforward solution is for the AI to have a mechanism through which it relinquishes control to the human. In such a way, a trained ATCo, for example would be able to take control of the situation and issue maneuvers that would ensure safety. Another example can be a version of the famous "trolley problem" where a drone, through some foreseen event, has to choose between to hit one of two objects, as it can't avoid both. A possible solution to these issues can be for the AI to relinquish control to the human operator [20, 181]. In such instances, the human can use their judgement in the case of collision between two aircraft, or common ethical principle such as consequentialism [182]: that the right thing to do iswhatever leads to the best results. However, we argue that this is not enough for a fully autonomous system. Such systems must have the ability to learn from these difficult situations in order to internalize ethical principles that drive humans. However, such principles are notoriously hard to quantify fully. Therefore, we argue that alignment with human values as suggested in the previous subsections should be guiding research in autonomous ATM systems.

The quest for autonomy

While the concerns expressed in this section might seem far fetched, the inclusion of more automated systems and the quest for autonomy will require AI models that function not only in ideal environments, but are able to handle all sorts of noisy information and implicit goals. As such, all future directions presented in the previous section, can be thought of as going towards the same direction: autonomous agents. Therefore, in order to reach this goal, we believe that more so than the answers provided by this thesis, the questions raised by our work provide the backbone for realizing that goal. From uncertainties, to meaningful human control, the answers of those questions must build on top of each other to reach a level of automation where autonomous agents can operate alongside humans, and where finally such systems can be trusted and transparent.

BIBLIOGRAPHY

[1] Andrew Cook. *European air traffic management: principles, practice, and research*. Ashgate Publishing, Ltd., 2007.

[2] *The Establishment and Management of a State's Aviation Security Oversight System, part B (Doc 9734)*. Tech. rep. ICAO, 2006.

[3] ICAO. *Procedures for Air Navigation and Air Traffic Management Pans-atm Doc 4444*. ICAO, 2016.

[4] *ATFCM Operations Manual*. Tech. rep. Eurocontrol, 2019.

[5] Michael Kreuz and Michael Schultz. "Modelling Interactions in Complex Systems–An Air Navigation Service Provider Focussed Approach". In: (2014).

[6] Andrea Ranieri et al. "STREAM–Strategic Trajectory de-confliction to Enable seamless Aircraft conflict Management". In: *Proceedings of the 1st SESAR Innovation Days (SIDs), Brussels* (2011).

[7] FEĐA Netjasov and DUŠAN Crnogorac. "Assessment of safety performance indicators of future air traffic management system". In: *Proceedings of the XLIV Symposium on operational research (SYM-OP-IS)*. 2017.

[8] *Global Air Traffic Management Operational Concept (Doc9854 AN/458)*. Tech. rep. ICAO, 2005.

[9] Marsel Omeri, Ralvi Isufaj, and Romualdo Moreno Ortiz. "Quantifying Well Clear for autonomous small UAS". In: *IEEE Access* (2022).

[10] James K Kuchar and Lee C Yang. "A review of conflict detection and resolution modeling methods". In: *IEEE Transactions on intelligent transportation systems* 1.4 (2000), pp. 179–189.

[11] JE Kuchar and Ann C Drumm. "The traffic alert and collision avoidance system". In: *Lincoln laboratory journal* 16.2 (2007), p. 277.

[12] Mykel J Kochenderfer, Jessica E Holland, and James P Chryssanthacopoulos. "Next-Generation Airborne Collision Avoidance System". In: *LINCOLN LABORATORY JOURNAL* 19.1 (2012).

[13] *ACAS Guide*. Tech. rep. Eurocontrol, 2022.

[14] Marıa Consiglio et al. "ICAROUS: Integrated configurable algorithms for reliable operations of unmanned systems". In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE. 2016, pp. 1–5.

[15] *Separation Management in the U-Space: a key enabler to boost U3 services take-off*. Tech. rep. 2021. URL: https://bubbles-project.eu/wp-content/uploads/2021/03/BUBBLES-Workshop-Separation-Management-in-the-Uspace.pdf.

[16] Jarrett Larrow Marcus Johnson. *UAS Traffic Management Conflict Management Model*. 2020. URL: https://ntrs.nasa.gov/citations/20205002076.

[17] Robert Schultz et al. "Free flight concept". In: *Guidance, Navigation, and Control Conference*. DOI: 10.2514/6.1997-3677. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.1997-3677. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1997-3677.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Russell Stuart and Norvig Peter. *Artificial intelligence-a modern approach 3rd ed*. 2016.

[20] Stuart Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[22] CG Davis, JW Danaher, and MA Fischl. "The influence of selected sector characteristics upon ARTCC controller activities". In: *Contract No. FAA/BRD-301). Arlington, VA: Matrix Corporation* (1963).

[23] Richard H Mogford, Elizabeth D Murphy, and Jeremy A Guttman. "Using knowledge exploration tools to study airspace complexity in air traffic control". In: *The International Journal of Aviation Psychology* 4.1 (1994), pp. 29–45.

[24] Richard H Mogford et al. *The Complexity Construct in Air Traffic Control: A Review and Synthesis of the Literature.* Tech. rep. CTA INC MCKEE CITY NJ, 1995.

[25] William Pawlak et al. "A framework for the evaluation of air traffic control complexity". In: *Guidance, Navigation, and Control Conference*. 1996, p. 3856.

[26] Banavar Sridhar, Kapil S Sheth, and Shon Grabbe. "Airspace complexity and its application in air traffic management". In: *2nd USA/Europe Air Traffic Management R&D Seminar*. 1998, pp. 1–6.

[27] Brian Hilburn. "Cognitive complexity in air traffic control: A literature review". In: *EEC note* 4.04 (2004).

[28] Michael W Hurst and Robert M Rose. “Objective job difficulty, behavioural response, and sector characteristics in air route traffic control centres”. In: *Ergonomics* 21.9 (1978), pp. 697–708.

[29] David K Schmidt. “On modeling ATC work load and sector capacity”. In: *Journal of Aircraft* 13.7 (1976), pp. 531–537.

[30] Irene Vincie Laudeman et al. “Dynamic density: An air traffic management metric”. In: (1998).

[31] Parimal Kopardekar et al. “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*. 2008, pp. 15–19.

[32] Pierre Flener et al. “Air-traffic complexity resolution in multi-sector planning”. In: *Journal of Air Transport Management* 13.6 (2007), pp. 323–328.

[33] T Chaboud et al. “Investigating the air traffic complexity: Potential impacts on workload and costs”. In: *Bruxelles, Belgium: EUROCONTROL* (2000).

[34] GM Flynn, C Leleu, and L Zerrouki. “Traffic complexity indicators and sector typology analysis of us and european centres”. In: *EEC Note* 20 (2003).

[35] Zhuoxi Song et al. “Measurement of controller workloads based on air traffic complexity factors”. In: *CICTP 2012: Multimodal Transportation Systems—Convenient, Safe, Cost-Effective, Efficient*. 2012, pp. 1903–1914.

[36] Mingming Xiao et al. “ATCEM: a synthetic model for evaluating air traffic complexity”. In: *journal of Advanced Transportation* 50.3 (2016), pp. 315–325.

[37] Mauro Marchitto et al. “Air traffic control: Ocular metrics reflect cognitive complexity”. In: *International journal of industrial ergonomics* 54 (2016), pp. 120–130.

[38] Hasan Ayaz et al. “Cognitive workload assessment of air traffic controllers using optical brain imaging sensors”. In: *Advances in understanding human performance: Neuroergonomics, human factors design, and special populations* (2010), pp. 21–31.

[39] Monica Z Weiland et al. “Real time research methods: monitoring air traffic controller workload during simulation studies using electroencephalography (EEG)”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 57. 1. SAGE Publications Sage CA: Los Angeles, CA. 2013, pp. 1615–1619.

[40] Petar Andraši et al. “Subjective air traffic complexity estimation using artificial neural networks”. In: *Promet-Traffic&Transportation* 31.4 (2019), pp. 377–386.

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

[42] S Mondoloni. "Airspace Fractal Dimensions and Applications". In: *The 4th ATM Seminar, 2001*. 2001.

[43] Hong Jie Wee, Sun Woh Lye, and Jean-Philippe Pinheiro. "A spatial, temporal complexity metric for tactical air traffic control". In: *The Journal of Navigation* 71.5 (2018), pp. 1040–1054.

[44] David Gianazza and Kévin Guittet. "Selection and evaluation of air traffic complexity metrics". In: *2006 ieee/aiaa 25TH Digital Avionics Systems Conference*. IEEE. 2006, pp. 1–12.

[45] David Gianazza. "Smoothed traffic complexity metrics for airspace configuration schedules". In: *HAL* 2008 (2008).

[46] David Gianazza. "Forecasting workload and airspace configuration with neural networks and tree search methods". In: *Artificial intelligence* 174.7-8 (2010), pp. 530–549.

[47] Hongyong Wang et al. "Study on evolution characteristics of air traffic situation complexity based on complex network theory". In: *Aerospace science and technology* 58 (2016), pp. 518–528.

[48] Thimjo Koca. "Spatio-Temporal Regions in the Context of Aircraft En-route Tactical Conflict Resolution". In: (2020).

[49] Karl Bilimoria. "Survey of air/ground and human/automation functional allocation for separation assurance". In: *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2012, p. 5413.

[50] Jian Yang, Dong Yin, and Haibin Xie. "A reinforcement learning based UAVS air collision avoidance". In: *Proceedings of the 29th Congress of the International Council of the Aeronautical Sciences, St. Petersburg, Russia*. 2014, pp. 7–12.

[51] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. "Improvement of conflict detection and resolution at high densities through reinforcement learning". In: *Proceedings of the ICRAT* (2020).

[52] Han Wen et al. "Application of DDPG-based collision avoidance algorithm in air traffic control". In: *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*. Vol. 1. IEEE. 2019, pp. 130–133.

[53] Max Hermans. "Towards Explainable Automation for Air Traffic Control Using Deep Q-Learning from Demonstrations and Reward Decomposition". In: (2021).

[54] Marc Brittain and Peng Wei. "Autonomous separation assurance in an high-density en route sector: A deep multi-agent reinforcement learning approach". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 3256–3262.

[55] Marc Brittain, Xuxi Yang, and Peng Wei. "A deep multi-agent reinforcement learning approach to autonomous separation assurance". In: *arXiv preprint arXiv:2003.08353* (2020).

[56] Marc W Brittain and Peng Wei. "One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory". In: *AIAA Scitech 2021 Forum*. 2021, p. 1952.

[57] Duc-Thinh Pham et al. "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.

[58] Zhuang Wang et al. "Deep reinforcement learning based conflict detection and resolution in air traffic control". In: *IET Intelligent Transport Systems* 13.6 (2019), pp. 1041–1047.

[59] Peng Zhao and Yongming Liu. "Physics informed deep reinforcement learning for aircraft conflict resolution". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[60] Sheng Li, Maxim Egorov, and Mykel Kochenderfer. "Optimizing collision avoidance in dense airspace using deep reinforcement learning". In: *arXiv preprint arXiv:1912.10146* (2019).

[61] Joris Mollinga and Herke van Hoof. "An autonomous free airspace en-route controller using deep reinforcement learning techniques". In: *arXiv preprint arXiv:2007.01599* (2020).

[62] Ramon Dalmau and Eric Allard. "Air Traffic Control Using Message Passing Neural Networks and Multi-Agent Reinforcement Learning". In: ().

[63] Javier Alberto Pérez-Castán et al. "Design of an ATC Tool for Conflict Detection Based on Machine Learning Techniques". In: *Aerospace* 9.2 (2022), p. 67.

[64] Laia Garrigó et al. "Visual analytics and machine learning for air traffic management performance modelling". In: *Proceedings of the 6th SESAR Innovation Days, Delft, The Netherlands* (2016), pp. 8–10.

[65] Esther Calvo Fernández et al. "DART: A machine-learning approach to trajectory prediction and demand-capacity balancing". In: *SESAR Innovation Days, Belgrade* (2017), pp. 28–30.

[66] Pedro Garcıa, Ricardo Herranz, and José Javier. "Big data analytics for a passenger-centric air traffic management system". In: *6th SESAR Innovation Days, Delft, Netherlands* (2016).

[67] Matthias Kleinert et al. “Machine learning of controller command prediction models from recorded radar data and controller speech utterances”. In: *7th SESAR Innovation Days, Belgrade* (2017).

[68] Debora Gil et al. “E-pilots: A system to predict hard landing during the approach phase of commercial flights”. In: *IEEE Access* 10 (2021), pp. 7489–7503.

[69] Giulio Di Gravio et al. “Overall safety performance of the air traffic management system: Indicators and analysis”. In: *Journal of air transport management* 44 (2015), pp. 65–69.

[70] Andrew Cook, Seddik Belkoura, and Massimiliano Zanin. “ATM performance measurement in Europe, the US and China”. In: *Chinese Journal of Aeronautics* 30.2 (2017), pp. 479–490.

[71] Arnab Majumdar and Washington Y Ochieng. “Factors affecting air traffic controller workload: Multivariate analysis based on simulation modeling of controller workload”. In: *Transportation Research Record* 1788.1 (2002), pp. 58–69.

[72] Sylvie Athènes et al. “ATC complexity and controller workload: Trying to bridge the gap”. In: *Proceedings of the International Conference on HCI in Aeronautics*. AAAI Press Cambridge, MA. 2002, pp. 56–60.

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

[74] *Manual on Air Traffic Management System Requirements*. Tech. rep. ICAO, 2008.

[75] Clément Peyronne et al. “Solving air traffic conflict problems via local continuous optimization”. In: *European Journal of Operational Research* 241.2 (2015), pp. 502–512.

[76] Romaric Breil et al. “Multi-agent Systems for Air Traffic Conflicts Resolution by Local Speed Regulation”. In: 2016.

[77] Jerom Maas et al. “The effect of swarming on a voltage potential-based conflict resolution algorithm”. In: *7th International Conference on Research in Air Transportation*. 2016.

[78] Marko Radanovic, Miquel Angel Piera Eroles, and Thimjo Koca. “Adaptive Aerial Ecosystem Framework to support the Tactical Conflict Resolution Process”. In: (2017).

[79] 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”. In: *IEEE Access* 7 (2019), pp. 104528–104541.

[80] Anthony J Masalonis, Michael B Callaham, and Craig R Wanke. “Dynamic density and complexity metrics for realtime traffic flow management”. In: *Proceedings of the 5th USA/Europe Air Traffic Management R & D Seminar*. 2003, p. 139.

[81] *Capacity Optimisation in Trajectory-based Operations*. `http://www.cotton-er.eu`. Accessed: 14-01-2021.

[82] Richard J Trudeau. “Dover Pub., ed”. In: *Introduction to Graph Theory* (1993).

[83] Jongkwang Kim and Thomas Wilhelm. “What is a complex graph?” In: *Physica A: Statistical Mechanics and its Applications* 387.11 (2008), pp. 2637–2652.

[84] Matthias Dehmer et al. “Measuring the complexity of directed graphs: A polynomial-based approach”. In: *Plos one* 14.11 (2019), e0223745.

[85] Hermann Gruber. “Digraph complexity measures and applications in formal language theory”. In: *arXiv preprint arXiv:1111.5357* (2011).

[86] Mehran Ahmadlou and Hojjat Adeli. “Complexity of weighted graph: A new technique to investigate structural complexity of brain activities with applications to aging and autism”. In: *Neuroscience letters* 650 (2017), pp. 103–108.

[87] Bosiljka Tadić, Stefan Thurner, and Geoff J Rodgers. “Traffic on complex networks: Towards understanding global statistical properties from microscopic density fluctuations”. In: *Physical Review E* 69.3 (2004), p. 036102.

[88] Ioannis Korkontzelos, Ioannis Klapaftis, and Suresh Manandhar. “Graph Connectivity Measures for Unsupervised Parameter Tuning of Graph-Based Sense Induction Systems.” In: *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*. 2009, pp. 36–44.

[89] Igor Mezić et al. “Spectral complexity of directed graphs and application to structural decomposition”. In: *Complexity* 2019 (2019).

[90] Alain Barrat et al. “The architecture of complex weighted networks”. In: *Proceedings of the national academy of sciences* 101.11 (2004), pp. 3747–3752.

[91] Ralvi Isufaj, David Aranega Sebastia, and Miquel Angel Piera. “Towards Conflict Resolution with Deep Multi-Agent Reinforcement Learning”. In: *14th USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)*. 2021.

[92] Yong Liu et al. “Multi-agent game abstraction via graph attention neural network”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 7211–7218.

[93] Cristina Barrado et al. “U-Space Concept of Operations: A Key Enabler for Opening Airspace to Emerging Low-Altitude Operations”. In: *Aerospace* 7.3 (2020), p. 24.

[94] D Geister and B Korn. “Concept for Urban Airspace Integration DLR U-Space Blueprint”. In: *German Aerospace Center-Institut of Flight Guidance* (2017).

[95] Francisco Javier Saez Nieto. “Collision Risk Model for High-Density Airspaces”. In: *Risk Assessment in Air Traffic Management* (2020), p. 3.

[96] Sergio Ruiz, Javier Lopez Leones, and Andrea Ranieri. “A novel performance framework and methodology to analyze the impact of 4D trajectory based operations in the future air traffic management system”. In: *Journal of Advanced Transportation* 2018 (2018).

[97] Fedja Netjasov, Dusan Crnogorac, and Goran Pavlovic. “Assessment of the Future Air Traffic Management System Safety Performance using Network-based Simulation Model”. In: *Proceeding of the SESAR Innovation Days (SID 2018), Salzburg, Austria*. 2018.

[98] *Recommended Key Performance Indicators for Measuring ANSP Operational Performance*. 2015.

[99] FAA and Eurocontrol. *Comparison of Air Traffic Management-related operational performance U.S./EUROPE*. 2017.

[100] Cinzia Battistella et al. “Methodology of business ecosystems network analysis: A case study in Telecom Italia Future Centre”. In: *Technological Forecasting and Social Change* 80.6 (2013), pp. 1194–1210.

[101] John Gulding et al. “US/Europe comparison of ATM-related operational performance”. In: *Air Traffic Control Quarterly* 18.1 (2010), pp. 5–27.

[102] MA Bedau. *Downward causation and the autonomy of weak emergence. Principia 6 (1), 5–50, reprinted as “Downward causation and the autonomy in weak emergence,”*. 2008.

[103] Miquel Angel Piera et al. “A Socio-Technical Simulation Model for the Design of the Future Single Pilot Cockpit: An Opportunity to Improve Pilot Performance”. In: *IEEE Access* 10 (2022), pp. 22330–22343.

[104] Ralvi Isufaj, Thimjo Koca, and Miquel Angel Piera. “Spatiotemporal graph indicators for air traffic complexity analysis”. In: *Aerospace* 8.12 (2021), p. 364.

[105] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[106] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: *Scientific reports* 9.1 (2019), pp. 1–12.

[107] Clark John and Holton Derek Allan. *A first look at graph theory*. Allied Publishers, 1995.

[108] Vincent A Traag. “Faster unfolding of communities: Speeding up the Louvain algorithm”. In: *Physical Review E* 92.3 (2015), p. 032801.

[109] 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*. Vol. 131. FAA/Eurocontrol USA/Europe. 2016, p. 132.

[110] Ralvi Isufaj, Marsel Omeri, and Miquel Angel Piera. “Multi-UAV Conflict Resolution with Graph Convolutional Reinforcement Learning”. In: *Applied Sciences* 12.2 (2022), p. 610.

[111] Ilya M Sobol. “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates”. In: *Mathematics and computers in simulation* 55.1-3 (2001), pp. 271–280.

[112] Andrea Saltelli et al. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[113] Jun Tang. “Conflict detection and resolution for civil aviation: A literature survey”. In: *IEEE Aerospace and Electronic Systems Magazine* 34.10 (2019), pp. 20–35.

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

[115] Lucia Pallottino, Eric M Feron, and Antonio Bicchi. “Conflict resolution problems for air traffic management systems solved with mixed integer programming”. In: *IEEE transactions on intelligent transportation systems* 3.1 (2002), pp. 3–11.

[116] Jacco M Hoekstra, Ronald NHW van Gent, and Rob CJ Ruigrok. “Designing for safety: the ‘free flight’air traffic management concept”. In: *Reliability Engineering & System Safety* 75.2 (2002), pp. 215–232.

[117] Yang Yang et al. “Multi-aircraft conflict detection and resolution based on probabilistic reach sets”. In: *IEEE Transactions on Control Systems Technology* 25.1 (2016), pp. 309–316.

[118] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[119] David Silver et al. "Mastering the game of go with deep neural networks and Tree Search". In: *Nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961.

[120] Maria Huegle et al. "Dynamic input for deep reinforcement learning in autonomous driving". In: *arXiv preprint arXiv:1907.10994* (2019).

[121] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. "Determining Optimal Conflict Avoidance Manoeuvres At High Densities With Reinforcement Learning". In: *10th SESAR Innovation Days* (2020).

[122] Ryan Lowe et al. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *arXiv preprint arXiv:1706.02275* (2017).

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

[124] Vincent François-Lavet et al. "An introduction to deep reinforcement learning". In: *arXiv preprint arXiv:1811.12560* (2018).

[125] Richard Bellman. "Dynamic programming and stochastic control processes". In: *Information and control* 1.3 (1958), pp. 228–239.

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

[127] Peter Henderson et al. "Deep reinforcement learning that matters". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[128] David Silver et al. "Deterministic policy gradient algorithms". In: *International conference on machine learning*. PMLR. 2014, pp. 387–395.

[129] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[130] Thimjo Koca, Ralvi Isufaj, and Miquel Angel Piera. "Strategies to Mitigate Tight Spatial Bounds Between Conflicts in Dense Traffic Situations". In: ().

[131] Aaron Wilson, Alan Fern, and Prasad Tadepalli. "Using trajectory data to improve bayesian optimization for reinforcement learning". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 253–282.

[132] Nina Schefers et al. "A constraint programming model with time uncertainty for cooperative flight departures". In: *Transportation Research Part C: Emerging Technologies* 96 (2018), pp. 170–191.

[133] JC Van Rooijen, I Grondman, and R Babuška. "Learning rate free reinforcement learning for real-time motion control using a value-gradient based policy". In: *Mechatronics* 24.8 (2014), pp. 966–974.

[134] Marc Peter Deisenroth. "A survey on policy search for Robotics". In: *Foundations and Trends in Robotics* 2.1-2 (2011), pp. 1–142. DOI: 10.1561/2300000021.

[135] Junzi Sun, Jacco M Hoekstra, and Joost Ellerbroek. “OpenAP: An open-source aircraft performance model for air transportation studies and simulations”. In: *Aerospace* 7.8 (2020), p. 104.

[136] Romaric Breil et al. “Multi-agent systems to help managing air traffic structure”. In: *Journal of Aerospace Operations* 5.1-2 (2017), pp. 119–148.

[137] SESAR JU. “European Drones Outlook Study”. In: *Single European Sky ATM Research* November (2016), p. 93. URL: `http://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf`.

[138] Cristina Barrado et al. “U-space concept of operations: A key enabler for opening airspace to emerging low-altitude operations”. In: *Aerospace* 7.3 (2020), pp. 1–18. ISSN: 22264310. DOI: `10.3390/aerospace7030024`.

[139] Thomas Prevot et al. “UAS Traffic Management (UTM) Concept of Operations to Safely Enable Low Altitude Flight Operations”. In: June (2016), pp. 1–16. DOI: `10.2514/6.2016-3292`.

[140] Jianping Zhang. “UOMS in China”. In: *EU-China APP Drone Workshop* (2018), pp. 6–8. URL: `https://rpas-regulations.com/wp-content/uploads/2018/06/1.2-Day1_0910-1010_CAAC-SRI_Zhang-Jianping_UOMS-_EN.pdf`.

[141] *UTM - A Common Framework with Core Principles for Global Harmonization*. `https://www.icao.int/safety/UA/Documents/UTM\%20Framework\%20Edition\%203.pdf`. Accessed: 11.10.2021.

[142] *NASA Conflict Management Model*. `https://www.nasa.gov/sites/default/files/atoms/files/2020-johnson-nasa-faa.pdf`. Accessed: 11.10.2021.

[143] Marko Radanovic, Marsel Omeri, and Miquel Angel Piera. “Test analysis of a scalable UAV conflict management framework”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233.16 (Sept. 2019), pp. 6076–6088. DOI: `10.1177/0954410019875241`. URL: `https://doi.org/10.1177%2F0954410019875241`.

[144] Maria Consiglio et al. “ICAROUS: Integrated configurable algorithms for reliable operations of unmanned systems”. In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. 2016, pp. 1–5. DOI: `10.1109/DASC.2016.7778033`.

[145] Sally C Johnson, Alexander N Petzen, and Dylan S Tokotch. “Johnson_DetectAndAvoid Aviation 2017 Handout”. In: June (2017).

[146] Guido Manfredi et al. “Are You Clear About ” Well Clear ” ? To cite this version : HAL Id : hal-01859067 Are You Clear About ” Well Clear ” ?” In: (2018).

[147] Brandon Cook, Tim Arnett, and Kelly Cohen. “A Fuzzy Logic Approach for Separation Assurance and Collision Avoidance for Unmanned Aerial Systems”. In: *Modern Fuzzy Control Systems and Its Applications* (2017), pp. 1–32. DOI: `10.5772/68126`.

[148] Maria Consiglio et al. “Sense and avoid characterization of the independent configurable architecture for reliable operations of unmanned systems”. In: *13th USA/Europe Air Traffic Management Research and Development Seminar 2019*. 2019.

[149] César A. Muñoz et al. “Unmanned aircraft systems in the national airspace system: a formal methods perspective”. In: *ACM SIGLOG News* 3 (2016), pp. 67–76.

[150] Andrew Weinert et al. “Well-clear recommendation for small unmanned aircraft systems based on unmitigated collision risk”. In: *Journal of Air Transportation* 26.3 (2018), pp. 113–122. ISSN: 23809450. DOI: `10.2514/1.D0091`.

[151] Timothy W. McLain and Matthew O. Duffield. “A Well Clear Recommendation for Small UAS in High-Density, ADS-B-Enabled Airspace”. In: 2017.

[152] Hemil C. Modi et al. “Applying Required Navigation Performance Concept for Traffic Management of Small Unmanned Aircraft Systems”. In: 2016.

[153] Jiechuan Jiang et al. “Graph convolutional reinforcement learning”. In: *arXiv preprint arXiv:1810.09202* (2018).

[154] Michal Skowron et al. “Sense and avoid for small unmanned aircraft systems: Research on methods and best practices”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233.16 (2019), pp. 6044–6062.

[155] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. “Review of conflict resolution methods for manned and unmanned aviation”. In: *Aerospace* 7.6 (2020), p. 79.

[156] Josh Bertram and Peng Wei. “Distributed computational guidance for high-density urban air mobility with cooperative and non-cooperative collision avoidance”. In: *AIAA Scitech 2020 Forum*. 2020, p. 1371.

[157] Xuxi Yang and Peng Wei. “Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility”. In: *Journal of Guidance, Control, and Dynamics* 43.8 (2020), pp. 1473–1486.

[158] Jueming Hu et al. “UAS Conflict Resolution in Continuous Action Space Using Deep Reinforcement Learning”. In: *AIAA AVIATION 2020 FORUM*. 2020, p. 2909.

[159] Guido Manfredi and Yannick Jestin. “An introduction to ACAS Xu and the challenges ahead”. In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE. 2016, pp. 1–9.

[160] John Boaz Lee et al. “Attention models in graphs: A survey”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.6 (2019), pp. 1–25.

[161] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. “A survey and critique of multiagent deep reinforcement learning”. In: *Autonomous Agents and Multi-Agent Systems* 33.6 (2019), pp. 750–797.

[162] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[163] Petar Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

[164] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[165] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.

[166] Vinicius Zambaldi et al. “Relational deep reinforcement learning”. In: *arXiv preprint arXiv:1806.01830* (2018).

[167] Ke Shi et al. “A Distributed Conflict Detection and Resolution Method for Unmanned Aircraft Systems Operation in Integrated Airspace”. In: Oct. 2020, pp. 1–9. DOI: 10.1109/DASC50938.2020.9256477.

[168] Michael Mullins et al. “Dynamic Separation Thresholds for a Small Airborne Sense and Avoid System”. In: 2013.

[169] Florence Ho et al. “Improved Conflict Detection and Resolution for Service UAVs in Shared Airspace”. In: *IEEE Transactions on Vehicular Technology* PP (Dec. 2018), pp. 1–1. DOI: 10.1109/TVT.2018.2889459.

[170] Sally C. Johnson, Alexander N. Petzen, and Dylan S. Tokotch. “Exploration of Detect-and-Avoid and Well-Clear Requirements for Small UAS Maneuvering in an Urban Environment”. In: 2017.

[171] Artificial Intelligence Roadmap. “A human-centric approach to AI in aviation”. In: *European Aviation Safety Agency* 1 (2020).

[172] Anna Jobin, Marcello Ienca, and Effy Vayena. “The global landscape of AI ethics guidelines”. In: *Nature Machine Intelligence* 1.9 (2019), pp. 389–399.

[173] Alan FT Winfield and Marina Jirotka. “Ethical governance is essential to building trust in robotics and artificial intelligence systems”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2133 (2018), p. 20180085.

[174] Yue Wang and Sai Ho Chung. "Artificial intelligence in safety-critical systems: a systematic review". In: *Industrial Management & Data Systems* (2021).

[175] Francis Rhys Ward and Ibrahim Habli. "An assurance case pattern for the interpretability of machine learning in safety-critical systems". In: *International Conference on Computer Safety, Reliability, and Security*. Springer. 2020, pp. 395–407.

[176] Johannes Fürnkranz et al. "Preference-based reinforcement learning: a formal framework and a policy iteration algorithm". In: *Machine learning* 89.1 (2012), pp. 123–156.

[177] KOLA Ilir, Ralvi ISUFAJ, and Catholijn M JONKER. "Does Personalization Help? Predicting How Social Situations Affect Personal Values". In: (2022).

[178] Dylan Hadfield-Menell et al. "Cooperative inverse reinforcement learning". In: *Advances in neural information processing systems* 29 (2016).

[179] SESAR Joint Undertaking. *Automation in Air Traffic Management: Long-term Vision and Initial Research Roadmap*. Accessed: 01-08-2022.

[180] Nick Bostrom. *Superintelligence*. Dunod, 2017.

[181] Brian Christian. *The alignment problem: Machine learning and human values*. WW Norton & Company, 2020.

[182] Walter Sinnott-Armstrong. "An argument for consequentialism". In: *Philosophical Perspectives* 6 (1992), pp. 399–421.

Appendix A

ANALYSING AIR TRAFFIC COMPLEXITY IN HIGH DENSITY SUAS OPERATIONS

In this chapter, we present some relevant results achieved from applying the complexity indicators introduced in Chapter 3 in high density operations for sUAS[1].

A.1 Spatiotemporal Interdependencies for sUAS

The airspace graph is defined at a certain time as an undirected weighed graph where each drone represents a vertex on it. The edges of the graph will represent interdependencies between UAS, more precisely, they will represent the *time to closest point of approach* (tCPA). We decided to use a time-based framework as most of the works on UAS have a time-based approach for conflict management. A threshold *thresh* will be defined, so if the tCPA value between two aircraft is less than that value, there will be an edge between them. The smaller the tCPA below the threshold value, the higher would be the edge weight value. Another threshold, H, is defined as the tCPA value which would result in the maximum weight. If the tCPA is less than H, the weight of the edge would always be the maximum weight value. The weights are normalized between 0 and 1 and their formal mathematical definition is:

$$w_{i,j}(t) = \begin{cases} 1 \text{ if } tcpa_{i,j}(t) \leq H \\ 0 \text{ if } tcpa_{i,j}(t) \geq thresh_h \\ \frac{thresh - tcpa_{i,j}(t)}{thresh} \text{ otherwise} \end{cases} \tag{A.1}$$

Where $tcpa_{i,j}(t)$ is the tCPA value between aircrafts i and j at time t. The definition of the weights leaves the graph as an undirected graph. As the edges and their weights are defined at each time step, the graph is extended to the temporal domain, so we will be able to take into account information like the aircraft heading, as if two aircrafts are moving towards each other, the weight of the edge connecting them will increase over time.

Another usual way of computing the weights of the graph representing the air traffic is using the distance between aircraft instead of the tCPA. In Figure A.1 a comparison

[1]This is joint work with Javier Garcia Cañadillas and Marsel Omeri in the context of the former's master thesis

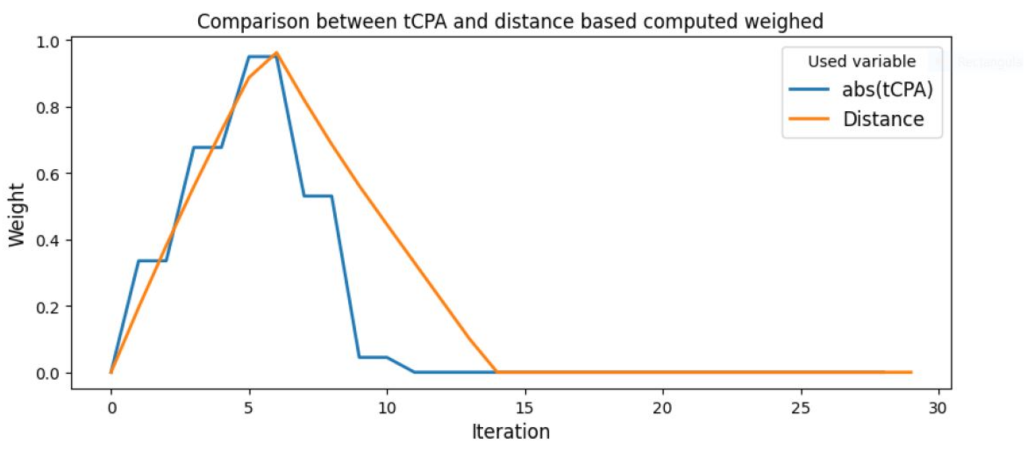


Figure A.1: Weights computed using tCPA and the distance between aircraft in a simulation consisting on two aircraft going towards each other. The velocities of the aircraft were 40 m/s, they were initially separated 200 meters. The thresholds for the tCPA weights were threshtcpa = 2.5 and Htcpa = 0, and for the ones computed using the distance were threshd = 200 and Hd = 0.

between the weights computed with tCPA and distance (as in Chapters 3 and 4) is shown. The situation is two aircraft going towards each other, beginning separated at a distance of 200 meters going at speeds of 40 m/s. The thresholds were chosen so both weighed were triggered in the beginning of the simulation and took their maximum values when the aircraft reach each other. In the computation of the weights, the absolute value of the tCPA was used. An advantage of using tCPA is that the value of the weights when decrease faster when the aircraft are moving away from each other than in the distance-based approach, as it can be shown in Figure A.1. This is an advantage of using tCPA for computing the weights because interdependencies should lose relevance when the aircraft are moving away as they can't no longer get in conflict with each other. Another benefit of using the tCPA is that it takes into consideration the speed of the aircraft, which is relevant to detect the severity of an intrusion and it is not contemplated by distance-based weights.

A.2 Experimental Setup

As with the rest of the work in this thesis, we used BlueSKy to simulate the scenarios and log the results. Each scenario consists of a number of aircraft that must go from one point in the airspace to the other, while passing through some waypoints. The chosen airspace was a circle with a radius of 0.5NM. The simulations were done varying two parameters: the number of aircraft (50, 70, 200) the tCPA threshold (25, 20 and 15 seconds). For each combination of parameters, 1000 total simulations were performed, half with conflict resolution (the CR algorithm used in this work was MVP). All the aircraft were generated at the same height (300 ft) in order to reduce the variability of simulations and in turn execution time. Based on the sUAS typical size and speed we have chosen 15 seconds for the look-ahead time for CD. The following paragraphs will explain the procedure to generate a scenario:

- **Waypoints creation**: a waypoint is a specific geographical point on a flight route that the aircraft is required to pass. In the beginning of the simulation, three waypoints (blue dots in Figure A.2) are generated: One at the centre of the circle and the other two aligned with it, separated from the centre by 10the waypoints are created is to force the aircraft to go through some common points so conflicts can be generated. If the UAS were generated with completely random headings, it would be very unlikely to observe any conflict.

- **Airspace initialization** a chosen number of drones are created with random positions inside the circle perimeter. Each aircraft trace a route with two points: the first one is one of the three created waypoints, chosen randomly, and the second one is one random point chosen from the circle perimeter. The aircraft first go towards the center of the circle where the waypoints are, and then return somewhere in the circle's perimeter. BlueSky directs aircraft through waypoints with the autopilot system LNAV, which stands for lateral navigation. The system measure the aircraft's position and compares it to the desired flight path. If the aircraft strays from the desired path, the system automatically makes corrections to keep the drone on track. The drones were generated with randomly selected speeds extracted from a uniform distribution between 10 m/s and 20 m/s.

- **Main loop**: the airspace began to be simulated inside the main loop, where all the necessary computations and logs are made and the airspace get updated right before the next iteration. The duration of the simulation was chosen depending on the radius of the circle. It was chosen so the slowest aircraft (with a speed of 10 m/s) could manage to travel a distance equivalent to the diameter of the radius.

The schematic flowchart of the simulation process is shown in Figure. BlueSky managed all the operations related to conflict detection and resolution and was also

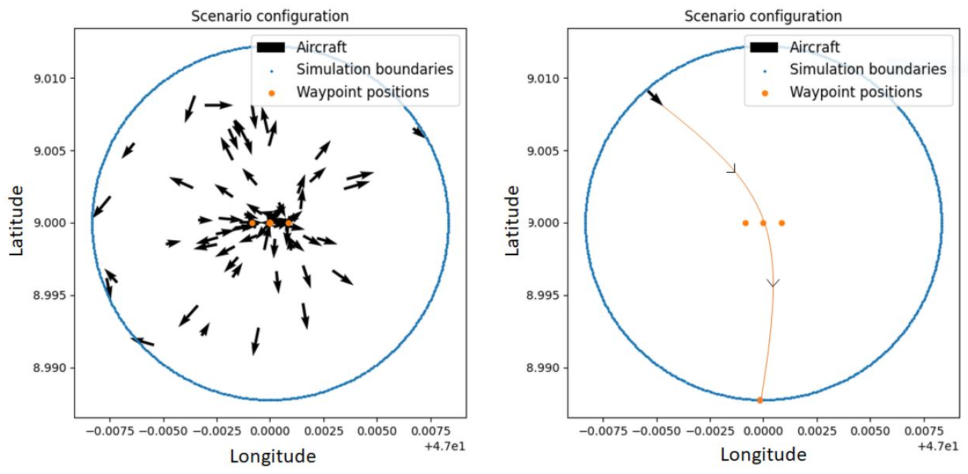


Figure A.2: In the first image of the airspace configuration in the middle of a simulation is shown. In the second one, an example of the path that an aircraft goes by in a simulation is displayed.

responsible of updating the state of the scenario after each iteration i.e. updating the positions and velocities of every aircraft based on their routes or the MVP instructions (if it is active). The rest of the computations were:

- **Number of drones control**: to keep the number of aircraft constant, at every timestep a function checks if there is any aircraft beyond the circle limits. If so, this aircraft is erased and another one is randomly created inside the circle's perimeter in the same way that the original ones were created.
- **Graph creation**: at each timestep, two different graphs were created using a Graph object from the Networkx[2] library in Python. In both the nodes were the aircraft at that timestep and the difference was the way the edges were constructed:
  - *Complexity graph*: this is the main graph representing the airspace through which the complexity of the scenario is measured using the indicators. The tCPA was computed using BlueSky conflict detection module and then equation A.1 was applied to create the edges between nodes.
  - *Conflict graph*: this is an auxiliary graph to find the compound conflicts. It is created when a conflict is detected. The edges connecting aircraft

[2]`https://networkx.org/`

have no weights and they exist if there is a conflict between them. The way to compute the compound conflicts, as previously described, was to use Networkx library to find all the connected subgraphs in the graph. By selecting those with a size bigger than 2 we get the list of compound conflicts.

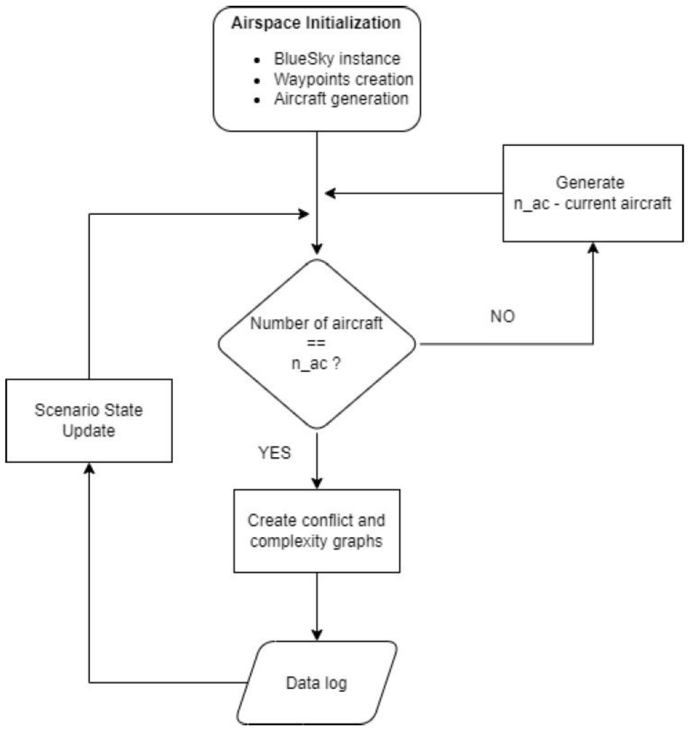


Figure A.3: Flowchart of the simulation code.

A.3 Results

In this section the simulation results are analysed. The code was parallelized, which results in a 10 fold speedup from the serial version.

Conflict distribution analysis

In this first part of the analysis of the results, a study of the amount and characteristics of the conflicts detected in the simulations has been done. Also, the effect that the resolution method has had on this values is studied, as well as the correlation between the conflicts and the number of aircrafts.

Effect of CR

In the logged data we can find the number of pair-wise conflicts, the number of Losses of Separation, the amount of compound conflicts and their sizes (the size of the biggest compound conflict if several were detected). Figure A.4a shows the number of conflicts depending on the number of aircraft and MVP. It can be seen that the total number of pair-wise conflicts is slightly higher in the case where we are using the Conflict Resolution method, which might seem counterintuitive but has several explanations. The first remark on this come from the definition of conflict. As it was stated in previous sections, a conflict is declared when a Loss of Separation is predicted to happen within a certain look-ahead time. The goal of a CR method is to solve conflicts in order to avoid a LoS, but it won't prevent drones from going into conflicts. In Figure A.4b the total amount of detected Losses of Separation is plotted and it can be noticed how this number is lower when a conflict resolution method is used. The fact that we get a higher number of total conflicts when MVP is active in the simulations can be explained as follows: when a conflict is detected, both aircraft execute a manoeuvre in order to solve it. If the aircraft density is high, this can make the UAS to cause secondary conflict with surrounding drones. This increases the expected number of conflicts compared to the situation where MVP is not used, as the aircraft follows straight paths most of the time. This is a first indication of MVP poor performance in high-density airspace, as it is likely to create more conflicts after solving one.

We can also expect the number of compound conflicts to decrease when we are using a CR method. In Figure A.4c the percentage of time that we measure at least one conflict is shown. It can be seen that in the case without MVP this percentage is significantly higher. When no CR method is used, conflicts last till the involve drones are separated from each other beyond their safety distance. However, when MVP is used conflicts get solved as soon as they are detected, which is reflected in the low percentage of time during which conflicts are present. This also can be noticed in the amount of compound conflicts, as it can be seen in Figure A.4d. This number is significantly lower when a conflict resolution method is being used because conflicts are not prolonged long enough for their to become complex. For example, if two aircraft get into a conflict and a third one is going to get in conflict with one of them, a conflict resolution method would most likely solve the first conflict before the second one occurs, avoiding the creation of a compound conflict of size 3.

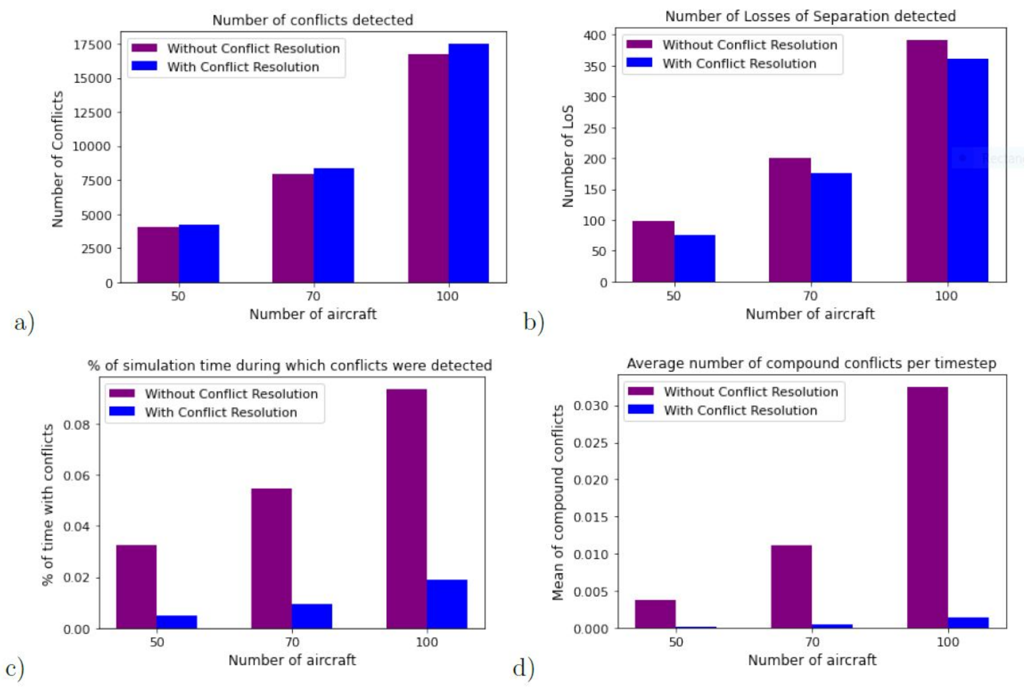


Figure A.4: Plots extracted from the study of the conflict distribution. In every figure, the purple color represents the simulations in which no conflict resolution method was applied while the blue color stands for those in which MVP was active. Total number of conflicts detected depending on the number of aircraft (a). Total number of Losses of Separation detected depending on the number of aircraft (b). Percentage of the simulation time during which conflicts were detected (c). Average number of compound conflict detected in every timestep (d).

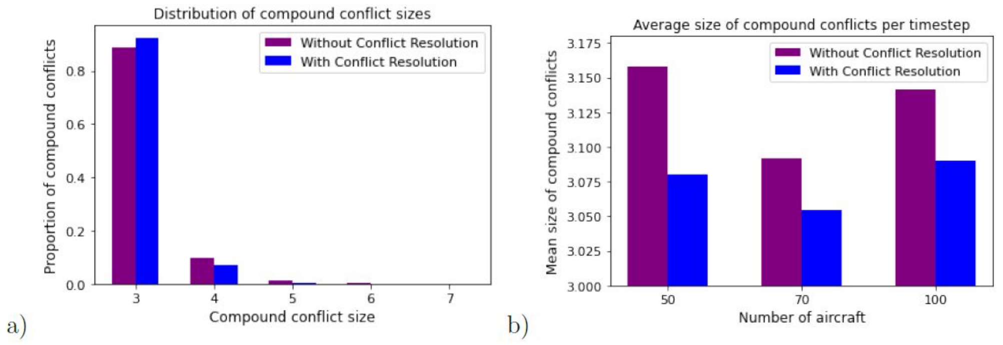


Figure A.5: Bar plot of the proportion of total conflicts that presented a certain size, separated by the present of Conflict Resolution methods in the simulations (a). Average size of the biggest compound conflict detected in every timestep depending on the number of aircraft (b).

Table A.1: Results of the Correlation test performed between different conflict characteristics (Pearson Correlation Coefficient and p-value)

	$n_{ac} \sim n_{conf}$	$n_{ac} \sim n_{compConf}$	$n_{ac} \sim \sqrt{n_{compConf}}$	$n_{ac} \sim confsize$
Corr. coeff (No CR)	0.99	0.98	0.99	-0.13
p-value (No CR)	0.06	0.10	0.02	0.91
Corr. coeff (With CR)	0.99	0.98	0.99	0.39
p-value (With CR)	0.06	0.11	0.04	0.74

One more difference that can be found in the conflict distribution between the simulations that use MVP and the ones that don't is the size of the compound conflicts. Figure A.5a, shows the distribution of the compound conflict sizes. It can be seen that the compound conflicts tend to be smaller when using MVP, for the same reason stated before: the conflicts get resolved before they get the chance to evolve into more complex structures.

Correlation between conflicts and UAS density

In Figures A.4 and A.5 the different variables has been shown as a function of the number of aircraft. The effect of the number of drones present the airspace on the occurrence and characteristic of conflicts has been studied. To do so, a Pearson Correlation test has been carried out between the number of aircraft n_{ac} and the number of pair-wise conflicts n_{conf} , the number of compound conflicts $n_{compConf}$ and the mean size of the compound conflicts confsize.

Looking at the figures, a monotonically increasing relation between n_{ac} and both $n_{compConf}$ and $n_{compConf}$ can be deduced. The Pearson correlation coefficient between these variables can be founded in Table A.1. For the number of conflicts we got a coefficient of 0.99 for both cases (with and without CR), which mean that there exist a nearly perfect linear relation. As we are dealing only with three values (the total number of conflicts per number of aircraft) we should attend at the p-value of the test to check how certain can we be about this. In this case the p-value is 0.06, which under a typical 0.05 significance level would make us reject the hypothesis of both variables being linearly related. However, given the low number of data used in this test, we will accept it as true. When looking at the correlation between n_{ac} and the number of compound conflicts we get a coefficient value of 0.98 with a p-value of 0.1. Again, a strong linear dependency is suggested by a Pearson coefficient value close to 1.0, but in this case the p-value is not small enough to accept it.However, it can be noticed in Figure A.4d that there might be a quadratic relationship between these variables, more than a linear one. To test this, we have repeated the Pearson test but taking the square root of $n_{compConf}$, because if they really follow a quadratic relation, taking the square root would cancel it out, leaving it a linear one. In this case, the correlation coefficient was 0.99 with a p-value of 0.02 for the case where no CR method is used and 0.04 for the case where MVP is active. We can accept the linear relation hypothesis under a 0.05 significance level, which allows us to accept as valid the idea that there is a quadratic relationship between the number of aircraft and the number of compound conflicts.

In Figure A.5b, the mean of the conflict sizes is shown. The correlation between nac and the size of the compound conflicts is not clear in this figure. A Pearson test reveals a correlation coefficient of -0.13 with a p-value of 0.91 for the case without CR and 0.39 with a p-value of 0.74 for the case with CR. No linear correlation can be deducted between these variables from these tests. This might be due to the fact that compound conflicts of sizes greater than 3 are very unlikely to happen, as it can be seen in Figure 13a, giving us very few data to study this relation.

Effect of CR on complexity

In this section the results on the study of the complexity in the simulations are shown. The changes in the complexity indicators depending on different factors as the use of MVP, the interdependencies time thresholds and the number of drones has been studied.

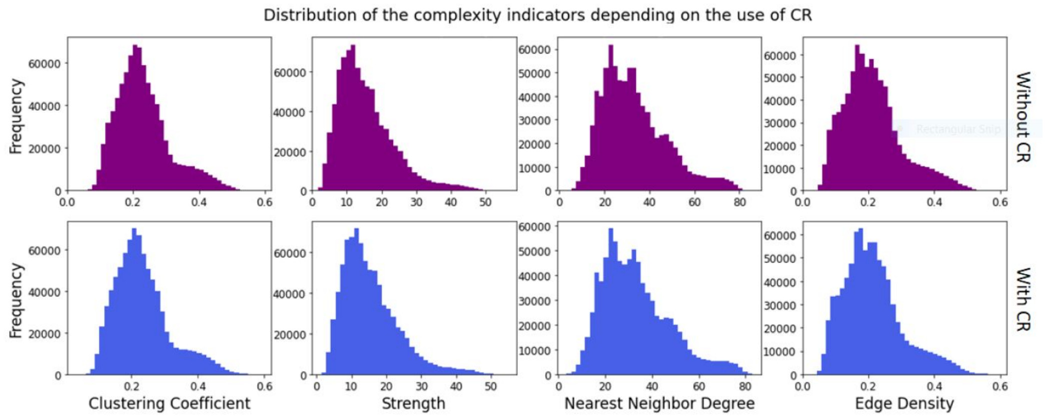


Figure A.6: Distributions of the complexity indicators separated by the use of a CR method. From left to right Clustering Coefficient, Strength, Nearest Neighbor degree and Edge Density are shown. The top row correspond to the simulations where no CR method was applied and the bottom row are the ones with MVP. The distributions were extracted by computing the histogram of each indicator.

One of the main objectives of this work was to study how the application of a conflict resolution method affects complexity in a high-density airspace. In this subsection, the results from this analysis are shown. The goal is to see whether the complexity indicators have higher or lower values when the MVP conflict resolution method is applied with respect to the case where it is not. To do so, we have extracted the histograms of each indicator to study their distributions. Figure 14 shows the distribution of each indicator separated into the cases where MPV is being used (second row) and when it's not (first row). Comparing each pair of distribution (column by column in the figure) nothing can be deducted at first sight. The indicator values seems to be very similar with some minor changes in the distribution shapes. In order to detect some difference we have needed to apply a statistical test named Wilcoxon-Mann-Whitney test, also known as U-test.

Usually to compare two distributions we would compare their means using a t-test, but this requires that the data follows a normal distribution and that it doesn't present any asymmetry. In our case, the complexity indicator distributions don't adjust to any of those requirements, so other test must be applied. The U-test is a non-parametric ranking test designed as an alternative to the t-test for this type of situations. It does not require the data to be normally distributed nor symmetric. The test goal is to look for any significant difference between two samples and it is based on the following: if both samples are similar and we add all the observations together and order them from smaller to larger, we would expect the observations of one and the other sample to be randomly interspersed. Otherwise, if one of the samples present lower or higher values than the other sample, when ordering the observations, they will tend to be grouped so that those of one sample are above those of the other. The null and alternative hypothesis that we are testing in this case are the following:

$$\begin{aligned} H_0 : P(NCR > CR) &= P(CR > NCR) \\ H_0 : P(NCR > CR) &= 0.5 \end{aligned} \tag{A.2}$$

$$\begin{aligned} H_a : P(NCR > CR) &> P(CR > NCR) \\ H_a : P(NCR > CR) &> 0.5 \end{aligned} \tag{A.3}$$

Where NCR and CR are the distributions of a certain complexity indicator for the cases where there are no conflict resolution method and for the ones there is, respectively. H_0 would be the probability that a randomly selected value from the indicator distribution when not using MVP being greater than a value extracted

Table A.2: Results of the U-test performed to compare the distributions of each indicator when using MVP and when not to test the effect of a conflict resolution method on the complexity of the airspace. The shown p-value corresponds to the probability of the null hypothesis from equations A.2 and A.3 being true. The alternative in this case is that the indicator values for the case with CR are higher than the ones for the cases without it.

	Clustering Coeff.	Strength	N.N. Degree	Edge Density
U-test statistic	3.470×10^{11}	3.472×10^{11}	3.473×10^{11}	3.471×10^{11}
p-value (CR >NCR)	0.003	0.01	0.03	0.004

from the distribution where we apply it is equal to the reciprocal statement. The alternative hypothesis Ha is these two probabilities being different.

In Table A.2 the U-tests results can be found. The p-value was computed to check if the values from the distributions using CR are higher than the ones from the distribution without it. As it can be seen, under a confidence level of 0.05 the null hypothesis can be rejected: the use of MVP actually increases the airspace complexity, contrary to what might have been expected. This might be explained the same way the increment of conflicts caused by MVP was explained. When the conflict resolution manoeuvre is executed by the aircraft in a high-density airspace, the aircraft are likely to increase the interdependencies with surrounding aircraft. This is also an indicator of the bad behaviour of MVP when it is applied in high-density scenarios as it doesn't take complexity into consideration, which can be concluded as a requirement for any CR algorithm to make sense in high-density airspaces.

Effects of the interdependencies threshold on complexity

The tCPA threshold value is a problem specific variable to be chosen. It is therefore important to study how the choice of the threshold affects complexity, so that the appropriate value can be chosen for each situation. Figure A.3 shows the average time evolution of each indicator over the simulations, separated by tCPA threshold. It can be seen that the higher the threshold value, the higher the complexity seems to be. If the threshold value is increased, more interdependencies will be created between aircraft and thus the situation will be more complex. The values in the figure show the mean of each indicator at every simulation timestep so we need a different representation to study the time when each indicator reaches its maximum and its value for each studied threshold.

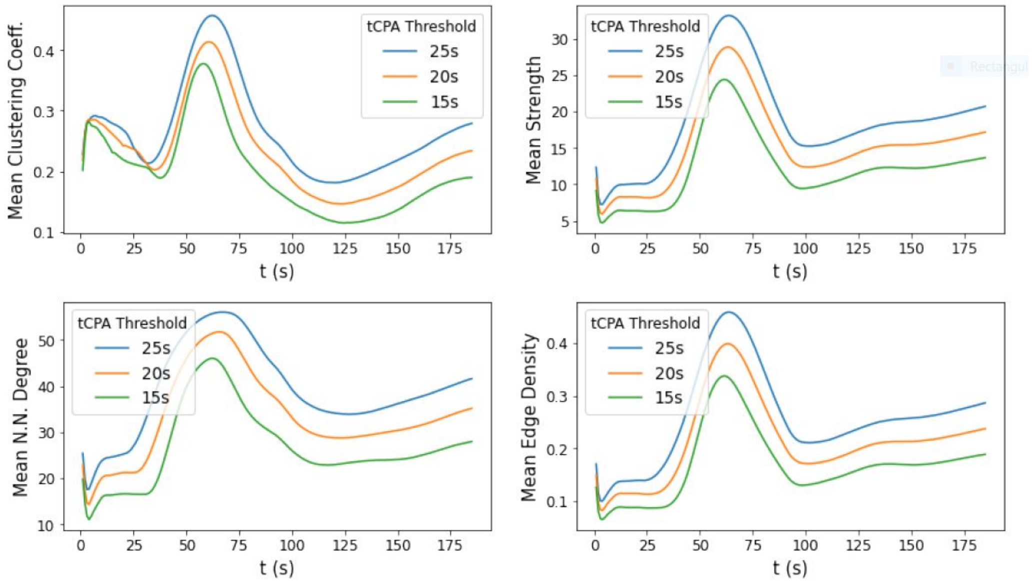


Figure A.7: Evolution of the complexity indicator separated by the tCPA threshold used in the scenario. Each plot shows the mean of the indicator for each simulation timestep when the simulations are filtered by tCPA threshold.

Figure A.8 shows a scatter plot of the maximum value reached by each indicator in every simulation and the simulation time when it occurs. The points have been separated by tCPA threshold. It can be seen that both the maximum complexity values and the time when they are reached tend to be higher with higher thresholds. The explanation for this could be that although interdependecies are detected later for the smaller thresholds, their duration are shorter so the maximum is reached before.

Effect of density on complexity

In this section, the effect of the number of aircraft on the airspace complexity is studied. Intuitively we would expect the complexity of the system to increase with the number of aircraft. The Figure A.3 shows the average time evolution of each indicator over the simulations, separated by number of aircraft (nac). What is observed for the case of Strength and the Nearest Neighbors Degree is a clear increase in complexity with the number of aircraft, as the curves are on top of each other throughout the simulation.

This makes sense as these indicators are not normalized and their values increases naturally with the total amount of edges created between drones. However, the plots

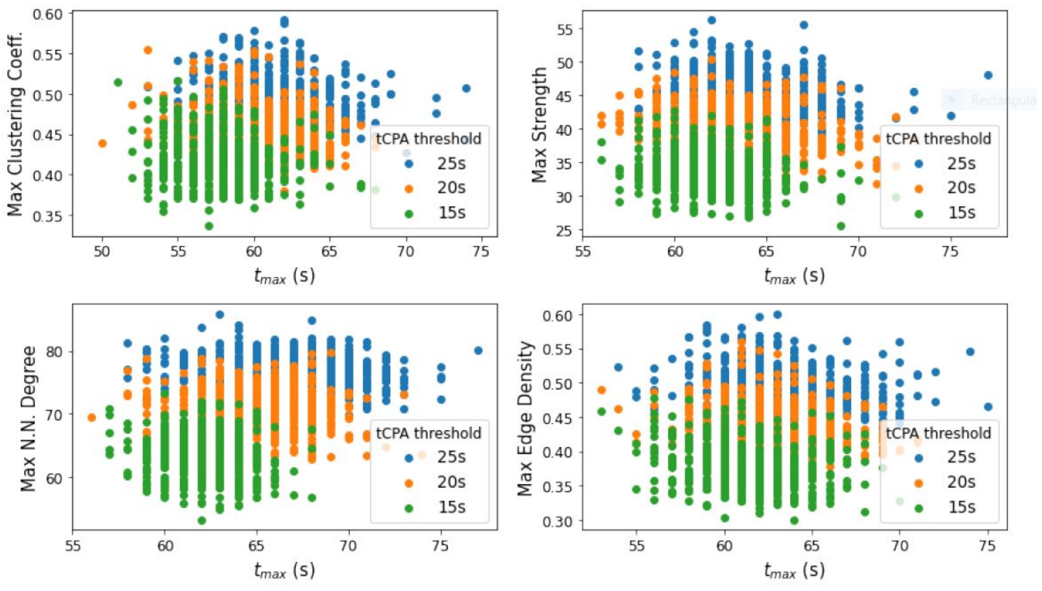


Figure A.8: Maximum value reached by each indicator in every simulation.

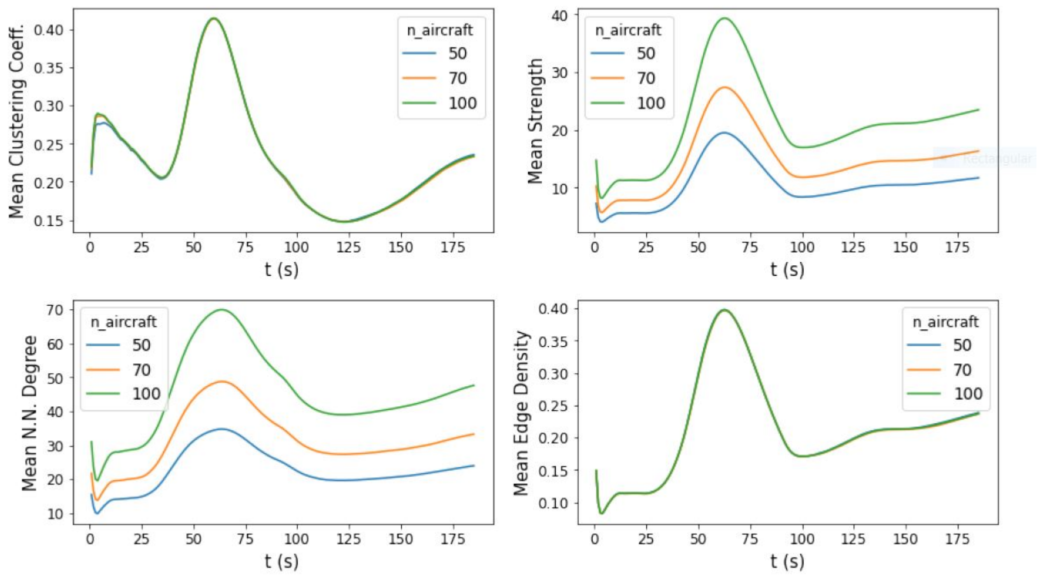


Figure A.9: Evolution of the complexity indicator separated by the number of aircraft in the scenario. Each plot shows the mean of the indicator for each simulation timestep when the simulations are filtered by number of aircraft. For Clustering Coefficient and Edge Density the curves are shown but they are mostly overlapped.

for the Clustering Coefficient and the Edge Density have their curves overlapped, indicating that there is no appreciable difference between the values of the indicators when varying the number of aircraft. Just like it was done to study the effect of

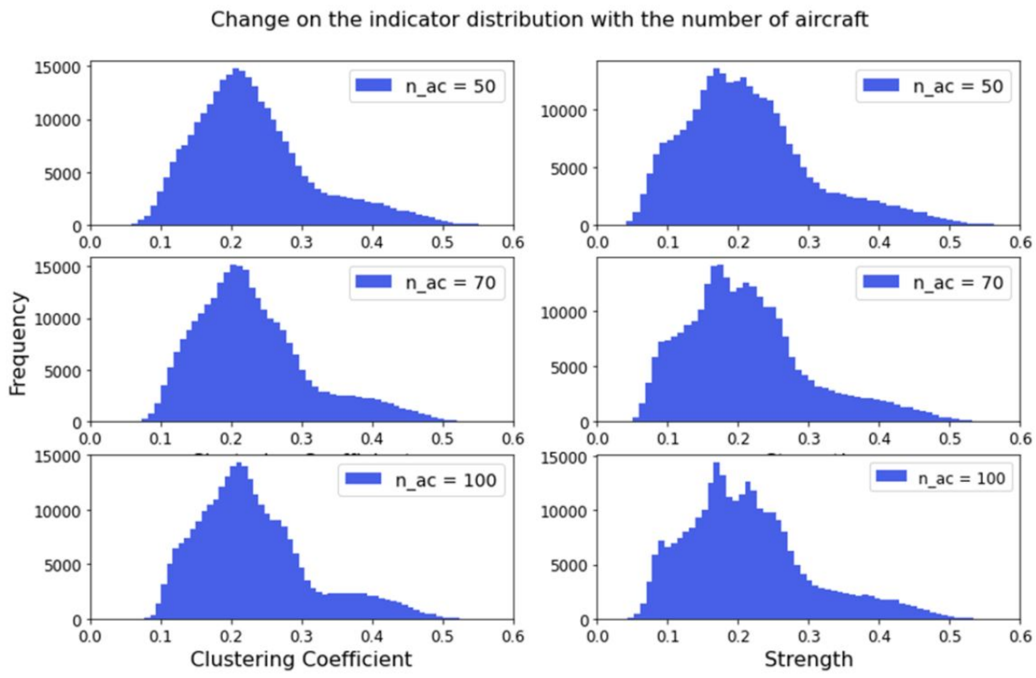
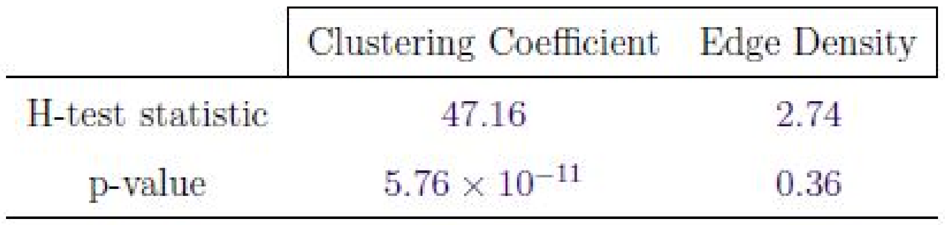


Figure A.10: Distributions of the Clustering Coefficient and the Strength when fixing the number of aircraft.

MVP on complexity, we have to apply a statistical test to the indicators distributions to find out if there is any meaningful difference between them. Figure A.10 shows the distribution of both indicators split by number of aircraft. No difference can be found at simple sight on this plots. Usually, to compare three or more distribution we would apply an ANOVA test, but it requires the data to be normally distributed and symmetric, which is not our case. An alternative to ANOVA that can be used in this situation is the Kruskal-Wallis test. The requirements for this test are that the data are ordinal and that all populations have the same type of asymmetry (rightward skewness in our case). It is a ranking method like the Wilcoxon-Whitney-Mann test: it orders all the data from lowest to highest and assign ranks to each point. If all samples come from the same population, the sum of the rankings of each sample should give similar numbers. The test measures the degree of deviation from this fact. The null H_0 and alternative H_a hypothesis that we are testing in our case are the following:

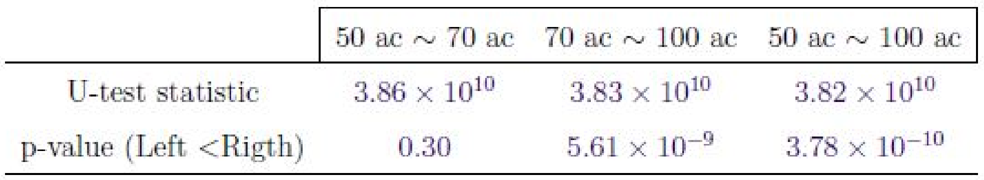
- H_0: All groups come from the same distribution and thus present similar values
- H_a: At least one of the three groups has a different distribution

Table A.3: Values extracted from the Kruskal-Wallis test or H-test performed on the distributions of the Clustering Coefficient and Edge Density when each on is separated in 3 groups according to the number of aircraft in the simulation. The p-value corresponds to the probability of the 3 groups to come from the same population and have similar values.



	Clustering Coefficient	Edge Density
H-test statistic	47.16	2.74
p-value	5.76×10^{-11}	0.36

Table A.4: Results of the Wilcoxon-Mann-Whitney test carried out on the 3 groups obtained by separating the distribution of the Clustering Coefficient indicator by number of aircraft. The test has been carried out pair-wised to find out if there was any difference between some of the groups.



	50 ac $\sim$ 70 ac	70 ac $\sim$ 100 ac	50 ac $\sim$ 100 ac
U-test statistic	3.86×10^{10}	3.83×10^{10}	3.82×10^{10}
p-value (Left <Rigth)	0.30	5.61×10^{-9}	3.78×10^{-10}

Performing the test on each of the 3 groups that can be seen in Figure 13 we get the statistic and p-value that can be found in Table A.3. As we can see, the p-value for the Edge Density case is high enough for the null hypothesis Therefore, the values of the Edge Density are similar regardless the number of aircraft in the simulation. The number of aircraft does not affect this complexity indicator. As it has been explained, the Edge Density is the ratio between edges in the graph and the total number of edges that the graph would have if it were a fully connected graph. The test results shows that this ratio does not change with the amount of aircraft.

However, the p-value for the Clustering Coefficient case is of the order of 10^{-11}, which indicates that at least one of the three groups present different values from the rest of the groups. In order to find which groups are different from each other, we have performed a Wilcoxon-Mann-Whitney (U-test) between the case with 50 and 70 aircraft, the case with 70 and 100 and the case with 50 and 100. The results of these test can be found out in Table A.4. The p-value indicates the probability of each pair of groups to have similar values while the alternative hypothesis is the first group having lower values than the second one. When comparing the 50 and 70 aircraft distributions we get a p-value of 0.30, which allows us to accept the null hypothesis and say that the Clustering Coefficient has similar values in these two groups. But, when comparing the 70 and 100 and the 50 and 100 we obtained p-values of the order of 10^{-9} and 10^{-10} respectively. The conclusion extracted from these results is that the complexity measured through the Clustering Coefficient does not change significantly when we have less than a 100 drones. However, for a large enough number of aircraft we obtain higher values for the Clustering Coefficient, indicating that the UAS tend to cluster more in this situation.

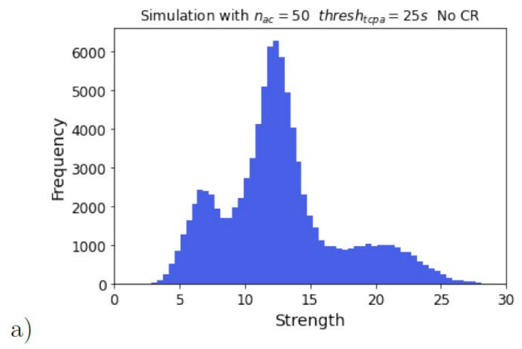
We can relate this last conclusion with the results we obtained when measuring how the number of compound conflicts changed with the amount of aircraft, as they can also be used as an indicator of clustering in the airspace. If this relationship had been linear, we would have expected the difference in Clustering Coefficient values to be either undetectable between 50 and 70 and 70 and 100 aircraft or noticeable between the two pairs of groups. But, as the relationship turned out to be quadratic, it makes sense that the difference in clustering between groups become more noticeable as the number of aircraft increases.

Complexity evolution in time

Figures and show how complexity evolves over time. It can seen the moment when the aircraft are grouped together in the centre of the stage, reaching maximum complexity and then moving away, decreasing it. If we plot the distribution of the indicators fixing all the parameters that have been varied in the simulations (number of aircraft, interdependencies threshold and the use of MVP) we obtain something like Figure A.11a, where the histogram of the Strength is shown for the case where $n_{ac} = 50$, $thresh_{tCPA} = 25$s and no CR is being used. What can be seen in the figure is how this distribution appears to be made up of three different distributions, as if three different cases can be distinguished in the simulation.

The study of this phenomenon has revealed that every simulation can be decomposed into three different stages. In Figure A.11b, the same histogram is computed but is filtered by some time intervals corresponding to each stage. It can be seen how the three independent distributions that give rise to the histogram in the Figure A.11a appears. The three different stages in the simulations and their estimated time intervals are:

- **t < 40s**: This first stage is the beginning of the simulation. All the aircraft



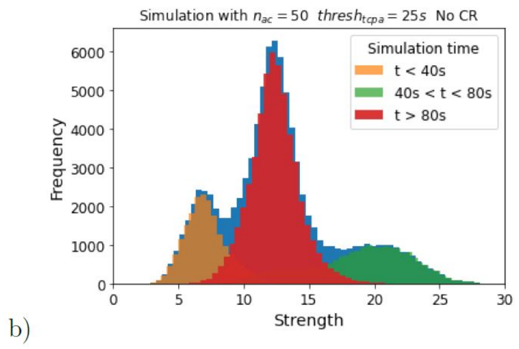


Figure A.11: Histogram of Strength for the case where n_{ac} = 50, $thresh_{tCPA}$ = 25s and no CR is being used (a). In (b) the same histogram is computed filtered by different simulation time to show the three stages that each simulation can be decomposed into. Each color indicates a different time interval.

are distributed along the circle perimeter and they are the most separated from each other that they will be in the simulation. This gives the minimum complexity, as it can be seen in the figure (orange distribution).

- **40s < t < 80s**: The second stage corresponds to the moment where all drones begin to gather in the center of the circle. Is the moment of the simulation when the aircraft are the closest as possible and the complexity increases up to its maximum (green distribution in the figure)

- **t > 80s**: After the aircraft pass through the waypoints located in the circle center, they go back to the perimeters. Meanwhile, others are created to substitute the ones that left the circle and it creates a situation where some aircraft are going towards the center and some of them are going away from the center. In this stage, the aircraft are more or less uniformly distributed across the circle and this results in intermediate values of the complexity (red distribution in the figure).