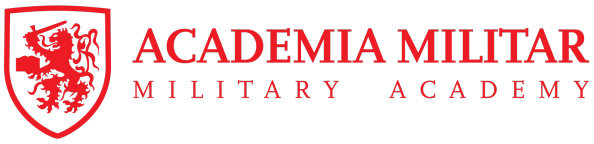


Anti-Jamming for UAVs using Artificial Intelligence based Swarm Behavior

Second Lieutenant Tiago Alexandre Sousa Silva

Thesis to obtain the Master of Science Degree in

Military Electrical Engineering

Supervisors:
Professor Doutor António Manuel Raminhos Cordeiro Grilo

Examination Committee
President:
Supervisor:
Members:

Lisbon, October 2024

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

"What we know is a drop; what we don't know is an ocean."
-Isaac Newton

Acknowledgements

This master's thesis has had significant contributions, allowing all the effort, determination, and perseverance invested in carrying out the work to be overcome with distinction and achieve the objectives.

To my supervisor, professor António Manuel Raminhos Cordeiro Grilo, a lecturer and specialist with vast experience in the field, with a scientific and academic curriculum of excellence and recognized merit, I express my deepest thanks for his support, patience, knowledge sharing and valuable contributions to the completion of this dissertation.

To my family, particularly my parents and brother, I express my gratitude for their exemplary encouragement and for never giving up or giving up at the least favourable times. I am deeply grateful for your help and understanding throughout this process and for everything you have given me and taught me throughout my life.

I want to say a special thank you to my girlfriend, Inês, for all her support throughout this master's thesis.

To my comrades at the Military Academy, especially the Foxtrot class, I would like to thank you for all your support during these six demanding years of training at the Military Academy, including the two years of training at the Instituto Superior Técnico and throughout my master's thesis. Without you, completing this journey would have been impossible, and it certainly wouldn't have been the same.

To the Military Academy, namely the teachers, instructors, commanders and everyone involved, I express my gratitude for the effort and dedication put into training and passing on knowledge to the students.

Finally, I would like to thank all those who, although they are not present, have made an equally significant contribution to my achievements.

I dedicate this master's dissertation to them all.

Abstract

The use of Unmanned Aerial Vehicles (UAVs) has seen remarkable growth in the performance of strategic tasks, both in civilian and military environments. This research, therefore, sets out to study the effectiveness of Anti-Jamming methods for UAVs supported by Artificial Intelligence based Swarm Behavior compared to conventional methods.

This work, therefore, provides an essential theoretical framework for understanding the proposed research, conducting a detailed study of the state of the art in the field of Unmanned Aerial Vehicles and Electronic Warfare. In addition, it highlights the fundamental tools required to carry out the research effectively, ranging from realistic UAV antenna models to advanced weapon intelligence techniques.

Several experimental tests and simulations were carried out to validate the hypotheses formulated, allowing concrete results to be obtained that demonstrate the effectiveness of the proposed strategies.

The results show that using a genetic approach can achieve effective results, although they are often costly regarding simulation time. In contrast, the Reinforcement Learning (RL) approach shows assertive results when pre-trained with realistic data in a helpful simulation time.

The results obtained not only corroborate the viability of the theoretical approaches developed but also provide valuable parameters for future research and practical applications in the field of defence and security.

Palavras-chave: Unmanned Aerial Vehicles, Anti-Jamming, Electronic Warfare, Comunications, Genetic Algorithms, Reinforcement Learning.

Resumo

A utilização de *Unmanned Aerial Vehicles* (UAVs) tem verificado um notável crescimento na realização de tarefas estratégicas, tanto em ambiente civil como militar. Por conseguinte, esta investigação propõe estudar a eficácia de métodos de *anti-jamming* suportados por *Swarm Intelligence* comparativamente aos métodos convencionais.

Este trabalho proporciona então um enquadramento teórico essencial para compreender a investigação proposta, conduzindo um estudo detalhado do estado da arte no domínio de *Unmanned Aerial Vehicles* e de *Eletronic Warfare*. Adicionalmente, destaca as ferramentas fundamentais requeridas para a execução eficaz da pesquisa, abrangendo desde modelos realistas de antenas de UAVs até técnicas avançadas de *swarm intelligence*.

Foram realizados diversos testes experimentais, divididos por cenários de diferentes graus de complexidade, e simulações para validar as hipóteses formuladas, permitindo a obtenção de resultados concretos que evidenciam a eficácia das estratégias propostas.

Os resultados mostram que a utilização de uma abordagem Genética pode alcançar resultados eficazes, embora sejam frequentemente dispendiosos em termos de tempo de simulação. Em contrapartida, a abordagem Reinforcement Learning (RL) apresenta resultados assertivos quando pré-treinada com dados realistas, para um tempo de simulação útil.

Os resultados obtidos não apenas corroboram a viabilidade das abordagens teóricas desenvolvidas, mas também fornecem parâmetros valiosos para pesquisas futuras e aplicações práticas no campo da defesa e segurança.

Palavras-chave: *Unmanned Aerial Vehicles*, *Anti-Jamming*, *Electronic Warfare*, Comunicações, Algoritmo Genético, Aprendizagem por Reforço.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACO Ant Colony Optimization.

AI Artificial Intelligence.

DRL Deep Reinforcement Learning.

DSSS Direct Sequence Spread Spectrum.

ESPAR Electrically Steerable Passive Array Radiator.

EW Electronic Warfare.

FANET Flying Ad-Hoc Network.

FHSS Frequency-Hopping Spread Spectrum.

GCS Ground Control Station.

GNC Graph Convolutional Networks.

ISTAR Information, Surveillance, Target Acquisition, and Reconnaissance.

MANET Mobile Ad Hoc Network.

MDP Markov Decision Process.

PPO Proximal Policy Optimization.

PSO Particle Swarm Optimization.

RL Reinforcement Learning.

SAC Soft Actor-Critic.

SIR Signal-to-Noise Ratio.

SNR Signal-to-Interference Ratio.

SPSP All-Pair Shortest Path.

SSSP Single Source Shortest Path.

UAS Unmanned Aerial Systems.

UAV Unnamed Aerial Vehicle.

UCAV Unmanned Combat Aerial Vehicle.

WN Wireless Network.

Mathematic Symbols

Parameters

E_b	Bit Energy
F_1	Free path loss in space
I_0	Power of interfering signal
J_g	Interference antenna gain
L_t	Transmission antenna losses
L_r	Receiving antenna losses
N	Number of time intervals
N_0	Power of thermal noise
P_j	Power of transmitted interference
P_t	Transmission power
P_{rj}	Instantaneous power of the received interference signal
P_{rt}	Instantaneous power of the received signal
$R_M[n]$	Achievable rate of last transmission
R_{g1}	Transmitting antenna gain
R_{g2}	Receiving antenna gain
S_j	Power of the interfering signal
S_t	Transmitted signal power

Chapter 1

Introduction

The continuous evolution of UAV technology has driven the creation of intelligent swarms, redefining how complex tasks are performed. These swarms made up of UAVs, have revolutionary potential by enabling cooperation between vehicles through inter-nodal communications.

However, like any other technology, it is exposed to agent attacks that affect its performance. Thus, this study enters the field of Electronic Warfare (EW), which develops the ability to exploit success in military, diplomatic and economic objectives. In military applications, EW provides the means to confront hostile actions involving the electromagnetic spectrum in all phases of battle. Thus, EW exploits the electromagnetic environment by detecting and analyzing the adversary's use of the spectrum and applying appropriate countermeasures.

1.1 Motivation

The application of UAVs has proven to be a highly flexible tool in various areas. The ever-increasing development of technologies supported by Artificial Intelligence (AI) has led to the consequent growth in the use of autonomous UAVs [1] in a variety of situations, namely: humanitarian aid in disaster situations [2], environmental control [3] and military application.

This work, supported by the relevance of the application of UAVs in a military environment, seeks to weaken enemy *jamming* capabilities in the face of a swarm of UAVs. Therefore, this research aims to explore the topic's relevance at a time when the application of these aerial vehicles as a weapon system has become so dominant and indispensable for acquiring advantage and destroying objectives in current armed conflicts [4].

The importance given to using these systems is based on the ability of UAVs, when used as communication nodes, to be highly mobile and used as a command receiver, data transmitter or communications relay. Combined with an increased capacity for concealment and autonomous control compared to conventional weapons systems, these characteristics make UAVs indispensable on contemporary battlefields [5].

In counter-response, these communication systems are the target of various jamming attacks seeking to hinder or eliminate their communication and coordination functions, leading to the reduction or even destruction of their operational capabilities. [6].

Currently, most Unmanned Aerial Systems (UASs) systems only feature anti-jamming techniques based on channel or frequency hopping. However, these techniques may not support data transmission at high speeds, either between an UAV and the earth station or between UAVs. However, transmission in these conditions is essential for coordinating the [7] swarm.

1.2 Objectives

The main objective of this master's thesis is to explore the application of realistic UAV antenna models and AI based swarm behavior techniques to circumvent the detected jammers, as well as to adapt the topology, actions, trajectories and traffic routing decisions between UAVs that minimize the effects of the blockage on communications. The work will develop suitable communication network simulation models and AI based swarm behavior (more specifically Genetic and Reinforcement Learning Algorithms) models, based on which system behavior will be optimized.

Thus, the proposed thesis aims to consider and study the actions that can be taken by a swarm of UAVs as an anti-jamming measure. Thus, the main question that this research aims to answer is the following:

- Can a swarm of UAVs, acting in coordination, adopt more effective techniques against jamming by jointly optimizing formation, antenna configuration and routing without compromising the mission?

The answer to this question is expected to provide a first step towards future development of hardware and software solutions that can be integrated in real UASs.

1.3 Contributions

This thesis makes several significant contributions to progress in UAV swarm communication research and anti-jamming strategies. The main contributions are described below:

A genetic algorithm-based solution was developed to optimise the positions and directions of UAV antennas, with the aim of maximising communication capacity in the presence of interference. This algorithm can be used in scenarios where it is necessary to find optimal communication configurations in real-time, with multiple variables involved.

Two RL algorithms were implemented, the Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) algorithms, which allow UAVs to learn optimal antenna movement and adjustment strategies to minimise the impact of interference from jammers. These algorithms allow for continually exploring new communication solutions in dynamic and complex environments.

An open-source library has been made available on GitHub, containing the code for the algorithms developed and the tools needed to simulate and optimise UAV swarm communication. Other researchers can use this library to replicate the results of this thesis, as well as to advance new research in the area of anti-jamming and optimisation of communication networks. The aforementioned repository can be accessed via the link: https://github.com/T1ago-S1lva/Anti-Jamm-with-Swarm-Intell.git.

These contributions not only advance the state of the art in the area of UAV swarm but also provide practical tools that other researchers can use to explore new solutions and further improve the resilience of communication systems in adverse environments.

1.4 Outline

This document is structured as follows. Chapter 1 introduces the primary motivation behind this thesis, explaining the problem it aims to solve: the use of Reinforcement Learning (RL) algorithms to optimise the swarm communication of UAVs in interference/jamming environments. The main contributions of this thesis, both theoretical and practical, are also outlined, highlighting the potential applications of the results obtained.

Chapter 2 addresses the state of the art related to the study topics. Fundamental concepts are explored, such as communication in UAV networks, anti-jamming techniques, and the main Reinforcement Learning algorithms applied to solving distributed network coordination and communication problems. In addition, existing approaches to dealing with interference in communication systems are analysed and how they compare to the proposal in this thesis.

Then, in Chapter 3 presents the structured formulation of the main problem, which involves the mathematical modelling of the communication system between UAVs and introducing jammers as adversarial agents. The communication model used is presented, which includes factors such as antenna gain, distance between UAVs, transmission power and interference from jammers. The Reinforcement Learning algorithm to be used is also discussed, detailing how the UAV control policy will be trained to optimise their positions and antenna orientation in order to maximise communication capacity while minimising the impact of interference.

Subsequently, Section 4 describes the experimental scenarios used to test the model and the proposed algorithms. These scenarios are organised by level of complexity, starting with controlled environments, where interference from jammers is limited, and moving up to more complex environments, where UAVs have to deal with multiple sources of interference on the move. This section explains the evaluation parameters that measure the system's performance, such as communication capacity, resilience to interference and the algorithm's convergence time.

Chapter 4 also presents and analyses the results obtained in the tests carried out. The performance achieved by the system in each scenario is discussed, focusing on metrics such as the average communication capacity between the UAVs, the bottleneck, and the efficiency of the anti-jamming strategies. In addition, a comparison is

made between the different system parameters and their impact on overall performance, offering a critical analysis of the proposed solutions and indicating possible limitations or points for improvement.

Finally, in Chapter 5, the main conclusions of this thesis are presented. This section summarises the contributions of the work, highlighting the most relevant findings and how they can be applied in real scenarios. Potential lines of future research are also identified, such as the application of more advanced RL algorithms or the integration of new machine learning techniques further to improve the resilience of the UAV swarm communication system.

Chapter 2

Background and Related Work

The development of this thesis requires knowledge *a priori* of certain concepts, which are essential for the development of the proposed research. These concepts are discussed in this chapter.

Initially, a reflection is made on the applicability and necessity introduced by the use of UAVs in the fulfilment of missions and the elaboration of military strategies and tatics.

Next, the an outline is provided regarding the vulnerabilities to which these weapons systems are subject and which technologies are currently used as a defence.

Subsequently, the need arises to explore and identify the characteristics and challenges encountered when using a swarm of UAVs as a communications system.

Once the topics above have been analyzed, this chapter introduces the training algorithms used and their suitability as a possible tool in the solution that this research proposes to achieve.

Next, the tools available to carry out the research are specified.

Finally, all the work related to this topic is presented, and the relevance of the research is summarized.

2.1 UAVs and their Military Applications

Recently, UAVs, individually or integrated into a network, have been widely exploited in military application contexts. Emerging communication technologies and high-performance remote control methods allow UAVs to present precise capabilities for exploiting an advantage over the enemy [8].

Depending on mission requirements, UAVs can be classified in terms of their characteristics and functions.

Reconnaissance and surveillance

UAVs can be used in an Information, Surveillance, Target Acquisition, and Reconnaissance (ISTAR) system to gather information about the enemy, locate targets and patrol hostile airspace without hostile airspace without endangering the lives of operators. In this way, collecting information obtained sensorially by the ISTAR increases combat potential and enriches the force's real-time information system about the battlefield [9].

Combat and Destruction of Objectives

Unmanned Combat Aerial Vehicle (UCAV) are UAVs equipped to hit assigned targets. UCAVs are capable of neutralizing and destroying targets on the battlefield with high precision. UAVs as a weapon system have been used successfully on battlefields such as Afghanistan, Pakistan and Yemen, where they have proven to be efficient and accurate. The use of UCAVs has become widespread and is currently used in most modern [9] operational environments.

Delivery of supplies

Still in a military context, but adjacent to civilian uses, UAVs can be used as a point source of supplies in certain locations on the battlefield [9]. Their remote control and independence from ground conditions make this a fast and safe form of refuelling.

2.2 Jamming and anti-jamming techniques

Although wireless technologies have developed rapidly in recent years, their ability to resist jamming attacks remains limited. Therefore, a careful analysis of jamming techniques and respective anti-jamming strategies is necessary.

2.2.1 Jamming

Jamming is an intentional and malicious action that uses radio interference to damage wireless communications by keeping the channel busy or corrupting the received signal.

The success of a jamming attack can be translated mathematically by the expression Signal-to-Noise Ratio (SNR), which represents the ratio between the power of the transmitted signal and the power of the interfering signal. It can then be expressed as follows:

$$SNR = \frac{P_r}{N_0 + I_0} \tag{1}$$

Where P_r is the power of the emitted signal, N_0 is the thermal noise and I_0 is the power of the interfering signal.

However, the effectiveness of malicious interference is subject to the robustness of the signal transmission mode. Metrically, the probability of a bit being received incorrectly can be expressed by the equation Bit Error Ratio (BER):

$$BER(SNR) = \frac{E_b}{N_0 + I_0} \tag{2}$$

Where E_b represents the bit energy. These attacks tend to affect the physical layer, but attacks between layers can be achieved [10]. The various types of interference will be described below [11]:

- **Spot Interference**

  This is the most common jamming method, where the jammer directs all of its power at a single frequency, with the same modulation as that used by the target, in order to cancel out the original signal.

- **Scanning interference**

  In this type of interference, the power used changes constantly between the various frequencies. This type of jamming can affect several frequencies, although not simultaneously. Although this method doesn't have the capacity to cancel out the original signal altogether, it can cause losses and retransmissions, consuming the target's energy resources and removing efficiency from established communications.

- **Barrage jamming**

  Unlike the two previous types of jamming, this method of jamming interferes with a range of frequencies simultaneously, which is its main advantage. Although with less power directed at the target, it can significantly reduce the SIR of enemy receivers.

- **Deceptive jamming**

  This type of jamming can be applied to a single or a set of frequencies where it invades the opposing Wireless Network (WN) with false data. In this way, unlike previous techniques, it bypasses the enemy's WN defence mechanisms. This technique will occupy the bandwidth intended for legitimate nodes.

2.2.2 Anti-Jamming

Various anti-jamming techniques have been developed as defense mechanisms against a jamming attack. This section will list and describe some of these techniques. The following techniques are explained in [11]:

- **Power Control**

  Given the directionality of a jamming attack, the attacker will initially have to identify the target before transmitting the malicious signal. In this way, protecting WN communications at a minimum power of use reduces the likelihood of discovery by an attacker. However, signals transmitted at a higher power are more resistant to a malicious signal but can also cause interference in friendly communications systems. Thus, signal power control is extremely important in defending against jamming attacks.

- **Frequency-Hopping Spread Spectrum (FHSS)**

  It consists of exchanging the radio signal carrier over several frequency channels. This topology requires an algorithm shared by the transmitter and receiver so that they can tune their frequencies, increasing the complexity of the system. Since a possible attacker won't have access to the shared algorithm, he won't be able to interfere effectively because the time window used by the carrier on the same frequency is reduced.

A drawback of this technique is that it requires more bandwidth than is necessary when using a single frequency.

- **Direct Sequence Spread Spectrum (DSSS)**

  The radio signal is encoded in this transmission type by multiplying a pseudo-random sequence of values 1 and -1. Subsequently, the receiver can recover the original signal by using *a priori* knowledge of this sequence. Since the result of this multiplication generates a signal similar to white noise, the attacker's task of detecting this signal becomes more difficult. In this technique we find the trade-off between spreading factor and bitrate.

- **Directional transmission**

  The application of directional antennas in WN can significantly improve the network's resistance to jamming attacks in contrast to omnidirectional antennas. The use of these antennas increases the gain by amplifying the signal in the antennas' privileged directions, consequently improving transmission performance and reducing the attacker's ability to interfere with communications: it becomes more difficult for the attacker to detect his target. The application of this technique is achieved through the use of smart antennas.

2.3 Smart Antennas and their Application in UAVs

Smart antennas, also known as adaptive antenna arrays, are antenna arrays that apply intelligent signal processing algorithms. This processing is used to extract signal characteristics such as direction of arrival and calculate beam vectors to locate the transmitter and direct signal transmission.

The use of smart antennas in communication systems provides several benefits and improvements over conventional antennas. By optimizing the direction of the transmission or reception beam, the communication system significantly increases performance by improving signal quality and reducing interference. This capability also promotes energy savings, often accounted for with caution, and security by significantly reducing the ease of opposing interception. This increased capacity has been proven in the [12] research.

In [13], an empirically tested Electrically Steerable Passive Array Radiator (ESPAR) was developed and was able to effectively control the direction of the radiation lobe and adapt it to the scaled frequencies. Therefore, directional communication capability will be the critical tool to enable the UAV swarm to circumvent a jamming attack effectively. However, as indicated in [13], a controller's presence is essential to direct the communication beam in the desired direction properly.

According to [14, 15], directional beamforming can be either digital or analogue. In their research, they proved that analogue beamforming shows relatively less flexibility, but a much lower operating complexity and low cost compared to digital directional beamforming.

2.4 Flying Ad-Hoc Network (FANET)

The FANET represents an innovative and dynamic category of wireless communication systems, where UAVs collaborate to form a temporary and autonomous network in the airspace. The dynamics of FANET networks present unique challenges for efficient communication between nodes, requiring specialized protocols and algorithms to overcome obstacles such as high mobility, variations in network topology and device resource limitations.

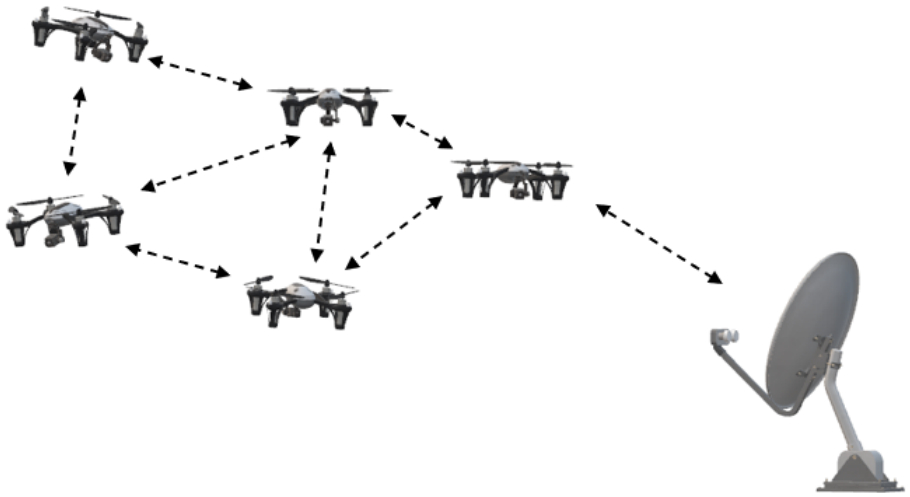


Figure 1: FANET, adapted from [16].

As depicted in Figure 1, this communications system shares data between the nodes of the swarm and between the swarm and the ground station (though the ground station is optional in a FANET).

Along with the evolution of embedded systems and the trend towards miniaturization of micro-electromechanical systems, the ability to produce small UAVs has developed. When individualised, these types of UAVs have limited capabilities. However, when integrated into an FANET, they have unique advantages [17]:

- **Scalability**

  The coverage provided by larger UAVs can become limited compared to the coverage achieved by a swarm of UAVs. When the communication system is implemented at the swarm level, greater fault tolerance and effective load distribution over each UAV is achieved. These strategies allow the communication system to be expanded without compromising performance, , although at the cost of some additional complexity.

- **Ability to complete the mission**

  When the operation is carried out by a single UAV, it is terminated when it fails. In the event of a failure in an UAV integrated into a swarm, the operation can continue. In addition to the advantage mentioned above, this control and communications system has the ability to coordinate task division between each UAV.

- **Radar Section**

  Unlike larger UAVs, smaller UAVs in a swarm have smaller radar-detectable sections, an essential feature in military operations.

However, a UAVs communications system has specific security vulnerabilities, given its distance from its control station, its dependence on coordination signals and the characteristics of the environment in which it operates. The following vulnerabilities stand out in a military application context:

- **Destruction or abduction of UAVs**

  UAVs in a FANET network are targets that are easily destroyed or susceptible to creating an inability to establish communications.

- **Dependence on a Ground Control Station (GCS)**

  The vulnerability of this communications system can be directly related to its dependence on the GCS to acquire indications about the mission to be carried out. As a result of the GCS being inoperable, the UAV communications system could lose its capabilities completely unless the swarm has instructions on how to operate autonomously or have the ability to operate autonomously.

- **Spoofing Attacks**

  Like the jamming attacks studied in this research, UAVs are also vulnerable to acquiring false data generated and emitted by an enemy force. This type of attack can insert false information about the location, identity, or mission to be carried out into the UAVs network.

- **Jammin against Coordination**

  It is common to apply jamming techniques to prevent coordination between UAVs, compromising the capabilities introduced by cooperativeness.

2.4.1 Routing protocols in FANETs

The effectiveness of FANET networks depends mainly on efficient routing protocols, which play a crucial role in ensuring reliable communications between UAVs. These protocols deal with the specific challenges associated with these dynamics of the networks and heterogeneous nature, including high mobility, variations in network topology and resource constraints of the UAVs. In the study carried out in [18], routing protocols are classified into four distinct groups: topology-based, geographical, hybrid and bio-inspired.

In topology protocols, the transmission must follow a path from the sender to the destination obtained according to the system's topology information. The network topology describes the physical or logical configuration of the interconnection between network nodes.

In [19], routing protocols based on network topology were studied, checking their effectiveness in terms of throughput, the delay generated and their response to a high network load. This research concludes by highlighting the protocols that perform best in high-mobility scenarios, the balance between network load and delay and, lastly, the one with the lowest delay.

As for protocols based on the geography of the nodes, they are designed to take into account the geographical information of the UAVs in the network to determine the most efficient routes for transmission between two nodes. These protocols use location data, such as geographical coordinates, altitudes or physical distances between nodes.

In the research described in [20], it was shown that the application of this type of routing protocol on a cooperative system achieved a higher transfer rate, lower delay, lower packet loss rate, lower transmission loss and lower network instability rate, through a fair retransmission strategy in order to select the most suitable and efficient route.

Hybrid protocols incorporate the mechanisms of the last two protocols mentioned to optimize route calculation.

The results of the application of this type of protocol are analyzed in [21], where it was able to overcome limitations identified in the topology protocols, namely eliminating iterative dependencies, resolving delays and providing optimization of the established routes, even through the unstable nature characteristic of these FANET networks.

Finally, bio-inspired protocols seek to resemble the natural phenomena found in communication between living beings. Examples include ant colonies, swarms of bees and even flocks of birds.

The research carried out in [22] demonstrates that the implementation of the ant colony-inspired routing protocol (Ant Colony Optimization) achieved the highest throughput and lowest delay from end to end, especially in conditions of high network load and high mobility, by choosing the best path for packet transmission.

2.4.2 Network Optimization

When the intention is to optimize an FANET network, considerations need to be made to find the lowest cost path.

The shortest path problem consists of a set of the issues associated with calculating the least-cost path between two vertices of a given graph. In the context of this thesis, these concepts will play a crucial role in optimizing communications in the swarm to minimize the energy expended for each transmission and the interference measured at the receiver.

The most relevant in the study of this thesis are [23]:

- All-Pair Shortest Path (SPSP), which consists of finding a path P between the source node (u_S) and the destination node (u_T) via the lowest cost path for all possible combinations of nodes.
- Single Source Shortest Path (SSSP), which consists of finding the shortest paths between a given vertex (u_v) and all other vertices in the graph.

Given the inherent complexity of research scenarios, the applicability of metaheuristics is relevant as high-level optimization strategies that guide other heuristics. In network optimization, metaheuristics help to find efficient solutions to problems such as routing, resource allocation, and balancing results with the required computing capacity. These problems usually involve a large number of variables and constraints, making them difficult to solve with exact methods in a reasonable amount of time. The following stand out:

2.5 Optimization Algorithms

Algorithm optimization is the process of improving the efficiency of an algorithm, both in terms of execution time and resource consumption, such as memory or energy. The aim is to reduce the computational cost, allowing algorithms to solve problems faster and more effectively, especially in scenarios where there are large volumes of data or critical time limits.

2.5.1 Genetic Algorithm (GA)

It simulates the process of natural selection, using operations such as selection, crossover and mutation to evolve potential solutions and find the best one.

GA starts by creating an initial population of individuals who can possibly solve the problem. Each individual is usually represented as a chromosome (a binary string or a string of numerical values) that encodes a possible solution to the problem. The creation of random chromosomes guarantees initial diversity in the population, which is crucial for effective exploration of the search space and prevents the algorithm from prematurely converging on a sub-optimal.

The population is a collection of several solutions, and the goal is to improve this population over several generations where each chromosome represents a possible solution to the problem. It can be a string of bits (for binary problems) or a sequence of values (e.g., integers, reals, etc.) that define the problem's parameters.

Each element of a chromosome is called a gene. Genes can represent the different variables or characteristics of a solution.

Then, the fitness function evaluates the quality of each chromosome. The idea is to assign a fitness value to each solution according to how well it solves the problem. GA tries to maximise or minimise this function over time, selecting and promoting the chromosomes with the highest fitness to generate the next generation. The fitness function must be carefully designed to accurately reflect the quality of a solution in the context of the problem.

GA applies three different operations:

- **Selection**

  It is the process in which the most adapted individuals are chosen to pass on their genes to the next generation. Individuals with greater fitness are more likely to be selected.

  Roulette Wheel Selection is where the probability of selecting an individual is proportional to its fitness value.

  Tournament Selection is where a random subset of individuals is selected and the individual with the best fitness among them is chosen.

  Elitism Selection ensures that the best individuals from one generation are kept for the next, preventing good solutions from being lost.

- **Crossover**

  Crossover combines two 'parent' chromosomes to create one or more 'children'. The idea is that children inherit characteristics from their parents, hoping that recombining good characteristics will lead to better solutions.

- **Mutation**

  Mutation is the random alteration of one or more genes on a chromosome. It introduces diversity into the population and prevents the algorithm from getting stuck in sub-optimal locations.

  The mutation rate is generally low (e.g. 1% or 5%), as widespread mutations can lead to unpredictable algorithm behaviour.

GA repeats each generation's selection, crossover, and mutation operations to create a new population. After each generation, the chromosomes of the latest generation are evaluated by their fitness function.

The process continues until a stop condition is reached, such as:

- Reaching a maximum number of generations.
- Finding a solution with sufficient fitness.
- No significant improvement in subsequent generations (convergence).

2.5.2 Particle Swarm Optimization (PSO)

It is an optimisation algorithm inspired by the social behaviour of groups of animals, such as flocks of birds, schools of fish or swarms of insects.

Based on the social behaviour of groups of living beings, where each particle (potential solution) moves in the solution space in search of the best position, influenced by its best experience and the best experience of the group. It is used to optimize network topologies and other configuration problems.

In this algorithm, each particle represents a potential solution and has a position in the (usually multidimensional) solution space, corresponding to a particular configuration of the problem to be solved. In addition to its position, each particle has a velocity that determines how it moves from one iteration to the next.

The particle's position defines the point in the solution space it is exploring and the velocity defines the direction and magnitude of the particle's movement in each iteration.

As with the Genetic Algorithm, the PSO uses a fitness function to assess the quality of each solution. The fitness function is used to measure how good the particle's current position is.

Each particle keeps a history of its personal best position, which is the position in which it has found the best solution (based on the fitness function) so far. The swarm also keeps a global best position, which is the best solution found by any particle in the swarm.

The velocity of each particle is updated based on three main components:

- **Inertia**

  The tendency of the particle to continue in the same direction it was previously moving.

- **Cognitive Component**

  The pull towards the personal best position (pbest) the particle has already found.

- **Social Component**

  The attraction towards the best global position (gbest) that any particle in the swarm has found.

The formula for updating the velocity of particle i in one iteration is given Equation 3.

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gbest - x_i(t)) \tag{3}$$

Where:

- $v_i(t)$ is the velocity of particle i at iteration t.
- w is the inertia factor, which controls the influence of the previous velocity. $c1$ and $c2$ are the learning coefficients for the cognitive and social components, respectively.
- $r1$ and $r2$ are random numbers between 0 and 1.
- $pbest_i$ is the personal best position of particle i.
- $gbest$ is the global best position of the entire swarm.
- $x_i(t)$ is the current position of particle i.

2.5.3 Simulated Annealing (SA)

Inspired by the annealing process in metallurgy, it is a metaheuristic that seeks an optimal solution by exploring the space of solutions in order to avoid local minima. In the metallurgical process, a material is heated to a point where its particles can move freely and then cooled slowly to allow the particles to organize themselves into a low-energy structure (optimal state).

Similarly, SA starts by exploring the solution space extensively (equivalent to high temperature) and gradually reduces this exploration (equivalent to cooling), allowing the solution to be guided to an optimal or near-optimal state.

The algorithm starts with an initial solution and a high initial temperature. The initial solution can be chosen at random or be a simple solution, depending on the problem.

The temperature is a control parameter that decreases over time. Initially, the high temperature allows a wide exploration of the solution space, accepting suboptimal or even worse solutions. This helps the algorithm escape from local minima.

As the temperature decreases, the algorithm becomes more selective, accepting only solutions that are better or comparable to the current one. At the end of the process, exploration is minimal, and the algorithm converges on the final solution.

The cost or energy of a solution is evaluated by an objective function, depending on the problem being optimised. The goal is to find a solution that minimises this cost function.

In each iteration, the algorithm generates a new solution (or state) by making a slight modification to the current solution. This modification is called a neighbour or perturbation of the current solution. Depending on the cost function of the new solution, the algorithm decides whether or not to accept it.

The cooling of the temperature is controlled by a decay function that defines how the temperature decreases over time. The choice of decay is crucial to the algorithm's performance. One of the most commonly used decay functions is exponential cooling.

$$T_{k+1} = \alpha T_k \tag{4}$$

Where α is a cooling factor between 0 and 1. A typical value for α is between 0.8 and 0.99.

If the temperature is cooled too quickly, the algorithm can get stuck in local minima. If it is cooled too slowly, the algorithm can take too long to find a solution.

2.6 Markov Decision Process (MDP)

Markov Decision Process is a model applied to decision making, used in control theory, decision theory and AI, supported by a set of actions and states, where the transition between states is made through the execution of actions. Thus, in an MDP, each agent present in a given state at a given time will have a set of actions that will make it transition from state to state according to its current state and objective.

Once the action has been carried out, a reward is given, positive or negative, depending on the new state and the action taken. The agent aims to maximize the total sum of rewards over time. This process is essential, as it is the only mechanism that guides the agent in its search for the best policy.

The Reinforcement Learning (RL) algorithms use MDP to model problems. An RL problem is formulated as an MDP so that sequences of decisions can be made to maximize the expected rewards over time.

- **Set of States (S)**

  The state fully describes the current condition of the environment. Each state $s \in S$ contains all the information necessary for the agent to know what is happening in the environment.

- **Set of Actions (A)**

  Actions are the choices the agent can make in each state. For each state $s \in S$, the agent can select an action $a \in A(s)$ that influences the transition to the next state.

- **State Transition Function (P)** The transition function defines the probability of moving from a state s to a new state s after performing an action a. It is formally represented by $P(ss, a)$, which is the probability that the next state is s, given that action a has been taken in state s.

- **Reward Function (R)**

  The reward function $R(s, a)$ defines the feedback the agent receives after taking an action a in a state s. The reward guides learning by incentivising actions that lead to desirable states.

- **Policy**

  The policy $\pi(a|s)$ describes the agent's behaviour, mapping states to actions. It defines what action the agent should take in each state.

- **State Value Function** Represents the expected value of the sum of future rewards that the agent can obtain from a state s, following a policy π.

  This is done using the Bellman Equation for the value function given by the Equation 5.

$$V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right] \tag{5}$$

  Where:

  - s is the current state .
  - π is the policy the agent is following.
  - $R(s_t, a_t)$ is the reward received for taking action a_t in state s_t.
  - $\gamma^t \in [0, 1]$ is the discount factor, which determines the importance of future rewards. A γ close to 1 gives more weight to future rewards.
  - $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$ is the total return weighted by the discount factor over time.

  The goal is to calculate the $V(s)$ values for all states $s \in S$ following the π policy.

- **Action Value Function**

  Represents the expected value of the sum of future rewards if the agent takes an action a in state s and follows policy π.

  Can be described by the Equation 6.

$$Q(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right] \tag{6}$$

  Where:

  - s is the current state .
  - a is the action taken in state s.
  - π is the policy that the agent follows after taking action a.
  - $R(s_t, a_t)$ is the reward received for taking the action a_t in state s_t.
  - $\gamma^t \in [0, 1]$ is the discount factor, which determines the importance of future rewards.

– $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$ is the total return weighted by the discount factor over time.

It is useful for learning the best action in each state, as the value of $Q(s, a)$ indicates how good action a is in state s, taking into account the expected future rewards.

2.7 Reinforcement Learning

Reinforcement Learning is an area of machine learning aimed at training an agent through rewards and punishments depending on the actions performed. Whenever an agent performs an action in a given state, it is assigned a reward where the agent's goal is to maximize the sum of the rewards.

In contrast to supervised learning, reinforcement learning does not need to train with previously evaluated actions to acquire the ideal behaviour it should try to follow for each action. In contrast, the training process rewards the model when it acts as intended and penalizes it otherwise. In this respect, reinforcement learning resembles human training, in which learning results from rewards and punishments in the process of trial and error [24].

Reinforcement learning algorithms are categorized into two distinct forms, model-based and model-free algorithms. In model-based algorithms, the agent models the environment it is in and, based on this model, predicts future rewards without having to carry out these same actions. In model-free algorithms, on the other hand, they don't need a model of the environment. They can only know the reward of a particular action by executing it, which means that model-free algorithms will repeat the same actions several times, updating their action strategy based on the outcome of their actions, to maximize the rewards [25].

Many RL algorithms have been developed in recent years, making it necessary to separate them into distinct classes. The most significant algorithms will be described.

2.7.1 Value-based

Value-based RL algorithms are methods that learn a value function, which estimates the expected reward from a state or state-action pair.

Q-Learning

This type of algorithm is based on the creation of an optimal function (Q function) between the action-state pair that estimates the expected reward for a given action. Iteratively, this algorithm updates its function in order to improve the estimates over time [26].

The algorithm starts with a table Q initialised with arbitrary values (usually zero) for each state-action pair.

Then use the Q function update, the Q(s,a) value is updated using the Equation 7.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \tag{7}$$

Where:

- s is the current state.
- a is the action taken.
- r is the reward received after performing action a.
- s is the new state after the action.

- α is the learning rate (how much new learning is applied).
- is the discount factor (considers future rewards).

The algorithm uses a strategy such as $\epsilon-greedy$ to balance the exploration of new actions and the exploitation of known actions. With probability ϵ, a random action is chosen; otherwise, the action with the highest Q-value is selected.

After sufficient iterations and visits to each state-action pair, the Q function converges to the optimal values, allowing the agent to determine the optimal policy.

2.7.2 Based on policy

Policy-based RL algorithms directly learn the policy, the function that maps states to actions, without having to explicitly model a value function.

Proximal Policy Optimization

PPO is an RL algorithm developed by OpenAI designed to be simple to implement and highly efficient in a wide range of tasks. PPO is a policy-based approach that focuses on optimising the policy directly rather than estimating a value function as in value-based methods.

PPO updates the policy through small iterative improvements, ensuring that the changes to the policy are not too drastic. This is achieved using a loss function that uses the probability ratio between the new and old policy.

The algorithm uses a modified objective function called the clip objective function. This function limits the amount that the policy can change in a single update, preventing excessive updates that can cause training instability. The Equation 8 gives the PPO loss function.

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{8}$$

Where:

- $r_t(\theta)$ is the odds ratio between the new and old policies.
- $\hat{A}_t$ is the advantage estimate.
- ϵ is a parameter that defines the clipping range.

PPO allows the use of mini-batches during training, enabling the algorithm to be more efficient and effectively use the data collected.

As is typical of RL algorithms, is necessary to adjust specific parameters, such as the number of *steps*, *batch size*, *learning rate* and *clip range*. These parameters can be adjusted by analysing the training performance metrics, namely:

- **Kullback-Leibler divergence approximation**

  This metric shows how divergent the new policy is from the previous one.

  A very high KL Divergence indicates instability in the agent-environment interaction process.

- **Entropy loss**

  This metric analyses the randomness with which the policy is studying the environment.

If the entropy is too low, the agent becomes too deterministic and does not explore new configurations.

This can lead to sub-optimal solutions. On the other hand, high entropy means that the agent is still exploring, which is desirable in the early stages of training.

- **Value loss**

  The value returned by this metric reflects how large the discrepancy is between the actual value and the predicted value.

  If the value loss is high, the agent has trouble correctly predicting each state's value. Monitoring this metric makes it possible to adjust the learning rate or other hyperparameters to improve the accuracy of the value predictions.

  Reducing the value loss helps the agent learn which actions are most advantageous.

- **Policy gradient loss**

  If the value is negative, this means that the policy is adjusting the actions in such a way as to increase the reward.

  If the policy gradient loss value is very close to zero, this may indicate that the policy updates are not having enough impact. On the other hand, very high values may indicate that the changes are unstable.

  Monitoring the policy gradient loss makes it possible to identify whether the agent is learning efficiently and whether actions result in better results.

According to the work developed in [27], policy-based algorithms are suitable for environments with large action spaces and high variance. Thus, in this research, the last two algorithms mentioned will be explored. These algorithms allow each agent to be accessible to acquire its policies when interacting with the environment through the application of neural networks.

2.7.3 Hybrids

Hybrid RL algorithms combine the principles of value-based and policy-based algorithms to get the best out of both.

Soft Actor Critic (SAC)

SAC is a popular algorithm in RL, off-policy, meaning it can learn from data not generated by the current policy. This feature allows it to efficiently reuse past experiences, which is helpful in complex and continuous environments such as this, where fine adjustments to UAV positions and antennas can significantly impact the quality of communication.

The SAC is designed to maximise the entropy of the policy, encouraging the agent to explore the environment and not limit itself to sub-optimal solutions, ensuring that the agent continually explores new possibilities for positioning and adjusting antennas.

As for its policy update strategy, this algorithm uses neural networks for both the policy (Actor) and the value functions (Critic). This update is achieved separately through the memory capacity introduced by the replay buffer.

In this way, the update of the Q-values in the critical networks is done through the loss function based on the difference between the estimated Q-value and the Q-target, which is calculated using the target networks and the

current policy. The calculation of the Q-value target is given by Equation 9.

$$y_i = r_i + \gamma\Big(\min(\text{Target Critic1}(s_{i+1}, a_{i+1}), \text{Target Critic2}(s_{i+1}, a_{i+1})) - \alpha \log \pi_\theta(a_{i+1} \mid s_{i+1})\Big) \tag{9}$$

The loss of the actor is defined by the Equation 10.

$$\text{Loss Actor} = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha \log \pi_\theta(a_i \mid s_i) - \min\left(\text{Critic1}(s_i, a_i), \text{Critic2}(s_i, a_i)\right)\right) \tag{10}$$

The actor network is responsible for learning and representing the agent's policy, i.e. it maps the observed states of the environment to actions. The goal of the Actor Network in SAC is to maximise the expectation of the accumulated reward and, simultaneously, maximise the entropy of the policy, encouraging the exploration of different actions.

In SAC, the policy is usually parameterised using a neural network that generates the actions for each state s. Actions can be continuous and suitable for continuous control tasks.

Critical Networks estimate the expected value of an action (Q-value) for a given state and action. In SAC, two critical networks are used to estimate Q-values, which helps avoid overestimating Q-values. These critical networks learn to predict the expected return of a given state-action pair $Q(s, a)$. Using two critical networks, Critic1 and Critic2, improves the robustness and stability of learning, as the minimum value between the two critics is used to update the policy. This avoids optimistic overestimations that can arise in other variants of Q-learning.

Target Networks are copies of Critical Networks but are updated more slowly and smoothly. The target networks aim to stabilise the training by providing a more stable reference for the updates of the critical networks. These networks are used to calculate the target values y_i, which are needed to update the critical networks. Instead of directly using the current criticals, which change with each update, the target criticals change slowly, resulting in more stable learning.

SAC uses a replay buffer to store the agent's previous interactions with the environment. This allows the agent to learn from past interactions, which is helpful in this problem where actions become complex.

Deep Q-Network (DQN)

DQN is a Deep Reinforcement Learning (DRL) algorithm that combines traditional Q-Learning with deep neural networks to deal with complex reinforcement learning problems, especially in high-dimensional environments such as games and simulations.

DQN extends traditional Q-learning by using a deep neural network to approximate the value function $Q(s, a)$ instead of using tables or other data structures that don't work well in large or continuous state spaces.

The neural network is used to approximate the Q function. Given a state s, the neural network returns an estimated $Q(s, a)$ for all possible actions a in state s. The DQN also uses a replay buffer to store previous state-action-reward transitions. The transitions are randomly sampled to train the neural network, which helps to break the correlation between consecutive samples and improves learning stability.

A copy of the main neural network is maintained and updated periodically. This target network calculates the value of $Q(s, a)$ for updating the main network, helping to stabilize the training as changes to the target network are slower.

Using deep neural networks, DQN can deal with problems where the state space is too ample to be stored or processed by Q-tables. The replay memory and the target network stabilize learning, solving common problems

such as temporal correlation and the oscillation of neural network parameters.

2.8 Development and Simulation Tools

The study of this research requires various tools to produce, test and analyses the results obtained realistically and with as few simplifications as possible.

Although there are tools dedicated exclusively to network simulation, such as ns-3, these require the creation of middleware between the simulator and the environment. The decision was therefore made to use the following tools to facilitate agent-environment interaction.

Annex A explains all the software packages used, their applicability, and their respective version.

Annex B contains the tables with the average simulation times recorded during all the stages of this thesis.

2.8.1 OpenAI Gym

It is an open-source library developed by OpenAI, designed to provide a standardized and flexible environment for developing the implemented RL algorithms. It is through these features that the different algorithms will be compared without having to change the environment under study.

Therefore, by carefully implementing this library, it is possible to define the environment in which our communications system is evaluated:

- Determining the distance between nodes.
- Determining the angular value between antennas of two UAVs.
- Implementation of the radiation diagram for directional antennas.
- Determining the transmit/receive gain of each antenna.
- Calculating the reception power.
- Calculating the interference power.
- Calculating the capacity between nodes.

2.8.2 Stable Baselines3

Stable Baselines3 (SB3) is a set of reliable and reproducible implementations of RL algorithms in PyTorch. Its compatibility with OpenAI Gym makes it possible to directly compare algorithms.

This tool is also characterized by its ability to receive extensions from the user and modify them according to the needs of the research in question.

Through careful parametrization of the algorithms provided by this library, we will be able to target and optimize the results obtained.

2.8.3 Seaborn

Seaborn is a data visualization library in Python. It provides a high-level interface useful for exploring and understanding complex data sets.

It is through the use of this library that we will be able to visualize, interpret and compare the results obtained.

2.9 Related Work

Different approaches to anti-jamming techniques have been studied, both individually and in groups. This chapter aims to list the most recent and most important advances in this research.

2.9.1 Graph Convolutional Networks (GNC) to predict the location and intensity of areas of interference

In [28], the use of GCN to predict the location and intensity of interference areas based on the information collected by each UAV is innovated. A multi-agent control algorithm is then used to disperse the swarm, avoid the interference areas and allow the UAVs to regroup when they reach their destination.

This research stands out for scenarios where the position of the jammer is unknown, focusing on predicting the areas of interference based on the collective intelligence of the UAVs, making it more robust and applicable to real scenarios.

The article concludes by demonstrating the effectiveness of the approach through simulations, where the UAVs could avoid interference and achieve their objectives efficiently.

2.9.2 RL in the multi-parameter adaptation of a UAV swarm

The article [29] deals with improving the performance of the UAV communications system in the presence of a jamming signal. The article proposes using multiple parameter programming in an RL algorithm to deal with this threat to swarm communications.

The communication system in question involves communication between a swarm of UAVs and a ground station that is subject to interference from various disturbing agents. Similar to what is proposed in this research, the relay in the system can optimize the quality of communication in the receiving area by exploiting different parameters, such as the frequency, movement and spatial domain of the antenna.

The proposed algorithm is a modification of the Q-Learning algorithm, adapted to optimize multiple parameters. A cost metric is introduced to balance the movement and communication performance of the UAVs. Simulation results demonstrate the effectiveness of the proposed algorithm in improving UAV swarm communication under interference.

2.9.3 RL in the topology control of an MANET

The acquisition of an alternative route between nodes in a network of UAVs, to avoid interference, becomes indispensable in our research.

The study at [30] addressed advanced anti-jamming strategies in mobile communications using intelligent radio devices. Focusing on the dynamics of the jammer, which can adapt its interference policy based on the security countermeasures implemented. In this study, mobile devices exploit the user's two-dimensional mobility to deal with interference. The effects of introducing specific parameters into a Q-learning algorithm were therefore investigated. The parameters selected for analysis were the frequency and mobility of individual nodes. Applying these parameters to the RL algorithm resulted in a significant acceleration of the iterative process and greater resistance to interference.

In [29], the problem of defense against jamming in a multi-agent scenario is addressed, taking into account coordination between users. The article uses a Markov game framework to model and analyze the problem of defense against jamming, proposing a multi-agent collaborative algorithm against jamming to obtain the optimal strategy.

The use of RL models was used to reduce the effectiveness of a jamming attack. In this way, the transmission directionality of each agent was explored, where the simulation results show that, compared to detection-based methods and independent Q-learning methods, the proposed model has the highest normalized rate. However, this study was designed for two elements only.

2.9.4 DRL in adjusting the frequencies used

The dynamic adjustment of frequencies plays a crucial role in anti-jamming strategies to protect communications systems from intentional interference, such as jamming attacks.

In the work [31], DRL models were employed to optimize frequency selection in heterogeneous high-frequency environments. The goal is to maximize the Signal-to-Interference Ratio (SNR) at the receiver. This approach aims to enhance spectral efficiency and improve communication performance by dynamically adapting the frequency of the high-frequency band based on environmental data. In this study, the environment is characterized only by the spectral state and the channel gain.

This article [32] addresses the problem of interference attacks that can hinder the efficient use of spectrum by cognitive radios. The channel selection problem is modeled as a Markov decision process. It proposes a synchronous real-time Q-learning algorithm with on-policy policy based on wideband spectrum sensing and greedy policy to actively avoid interfered channels.

The results showed that channel selection based on the proposed algorithm achieves a higher packet success rate than selecting the best channel based on detection alone. The results are even better when the sending cognitive radio cooperates with the cognitive radio receiving the packets to detect the interference and update the Q values.

2.9.5 Cooperative multi-agent network

The ability to mitigate the harmful effects of malicious interference by improving the reliability and efficiency of communications in environments subject to jamming attacks becomes essential in this context. The cooperative diversity introduced by the proposed scheme plays a crucial role in overcoming interference and maintaining connectivity in challenging conditions.

The relevance of this approach in the context of anti-jamming lies in its ability to mitigate the harmful effects of malicious interference, improving the reliability and efficiency of communications in environments subject to jamming attacks. The cooperative diversity introduced by the proposed scheme plays a crucial role in overcoming interference and maintaining connectivity in challenging conditions.

In the study carried out in [33], a cooperative anti-jamming scheme was proposed and designed, introducing the notion of cooperative diversity. There are two levels of cooperation.

In the first, a cooperative channel access scheme is applied where the access probabilities for each user are optimally regulated so that users affected by malicious interference have an increased share of airtime.

At the physical layer, users are capable of improving another user's link capacity through cooperative transmission and collaborating as relays with a certain probability.

Thus, a distributed price-based algorithm was designed to jointly optimize these two levels of cooperation. It was shown that, compared to non-cooperative algorithms, it achieved considerable gains. In the results of this research, the gain increased with increasing network traffic and interference power and demonstrated significant cooperative gains when a network is under very low throughput.

2.9.6 Metaheuristics for route planning

In a scenario with multiple UAVs together, route planning allows each UAV to follow a path that maximizes coverage of the area of interest, reduces response time and minimizes energy consumption.

Route planning is important because it avoids collisions between UAVs and ensures efficient and continuous communication between UAVs in an intelligent manner. By taking into account factors such as obstacles, weather conditions and different initial energy levels of the UAVs, effective planning ensures that missions are carried out with greater precision, less risk of failure, and greater savings in resources, maximizing the performance of the swarm.

In [34], Particle Swarm Optimization (PSO) was combined with the Ant Colony Optimization (ACO) to form a hybrid algorithm. The idea was to take advantage of the fast global search capacity of PSO and the robustness and optimisation efficiency of ACO. The hybrid PSO-ACO algorithm showed more incredible convergence speed and better optimisation results than ACO alone. Specifically, the hybrid algorithm converged to an optimal solution in approximately 200 iterations, while ACO alone took around 350 iterations to converge.

In the article [35], route planning for UAVs in search and rescue missions is addressed. It implements a GA that encodes route solutions as chromosomes, which undergo selection, crossover and mutation processes to find the best combination of path and energy consumption. The fitness function evaluates each route based on the length of the path and the variation in energy factors, guiding evolution towards more balanced solutions. This approach optimized message delivery and reduced overall communication delay.

In [36] several GA were also applied to plan safe paths for UAVs in target coverage problems, integrating additional deviation points to avoid collisions with the terrain. The results showed a significant performance of the GA based on clusters centers formed specifically at collision points.

2.10 Summary of Literature Review

Looking at the literature mentioned above, we can reflect on the application of RL algorithms in 3 different fields: position acquisition, adapting the frequencies of the various links and cooperation between nodes in order to optimize the communications system.

In this way, we understand that the combined application of frequency targeting and spatial freedom has not yet been scientifically explored. There is also little research into anti-jamming techniques in a communications system made up of a swarm of UAVs.

The literature review is summarized in Table 1.

Reference	Context	Contribution
Graph Convolutional Networks (GNC) to predict the location and intensity of areas of interference		
[28] 2024	Use of GCN to predict the location and intensity of interference areas based on the information collected by each UAV	Accurate prediction of jamming areas without prior knowledge of the jammer's location. Use of spatial movements of UAVs to avoid interference. Reducing computational costs through multi-agent control
RL in the multi-parameter adaptation of a UAV exam		
[29] 2019	Adaptation of the antenna's frequency, motion and spatial dominance through Q-learning.	More effective communications under interference.
DRL for adjusting the frequencies used		
[30] 2018	Alternative path acquisition in UAV networks. Exploration of specific parameters in Q-learning.	Acceleration of the iterative process and greater resistance to interference.
[29] 2019	Defense against jamming in a multi-agent scenario. Use of RL models to reduce the effectiveness of the attack.	Coordination between users for effective anti-jamming strategies.
RL in the topology control of a MANET		
[31] 2018	Dynamic frequency adjustment in anti-jamming strategies. Use of DRL models to optimize frequency selection.	Improved SNR and dynamic adaptation to environmental conditions.
[32]	Channel selection in cognitive radios to combat interference attacks. Use of Q-learning in real time.	Improved packet success rate and cooperation between cognitive radios.
Cooperative multi-agent network		
[33] 2016	Cooperative anti-jamming scheme with cooperative diversity. Use of distributed price-based algorithm.	Significant increase in reliability, efficiency and cooperative gains in low throughput scenarios.
Metaheuristics in route planning		
[34] 2021	PSO-ACO applied to the vehicle route planning problem, specifically to solve vehicle routing problems	Increased efficiency, faster convergence and significant distance savings.
[35] 2020	Application of GA in order to obtain routes for rescue UAVs based on the balance between path and energy required.	Increased efficiency, reducing overall delay with lower energy consumption.
[36] 2021	Use of GA in the coverage of targets performed by UAV routes, avoiding collisions with the terrain.	It demonstrates the careful choice of detour points based on local collisions on UAV routes, ensuring safer and more efficient trajectories.

Table 1: Summary of the literature review

A review of the literature above shows that no approach is identical to the one proposed in this thesis.

Therefore, this thesis intends to apply the movement capacity reflected in the work [29], extrapolating from the relay to each swarm element.

To this capability will be added the exploitation of the direction of the directional antennas, individualised to each UAV in the swarm.

Finally, as an intermediate process, each action's route planning will be calculated to evaluate each topology realised by the agent.

Chapter 3

Developed Algorithms for Anti-Jamming using AI based Swarm Behavior

In this section is also specified the application of swarm intelligence techniques will be explored in order to improve the resilience of communication systems against malicious interference. Specific algorithms, optimization methods and strategies will be discussed. Strategies used to avoid and overcome interference caused by jamming attacks.

3.1 System Model

In this context, the proposed model exploits the directional properties of each UAV's antenna to establish links with less susceptibility to interference, together with the determination of strategic positions that favour these connections.

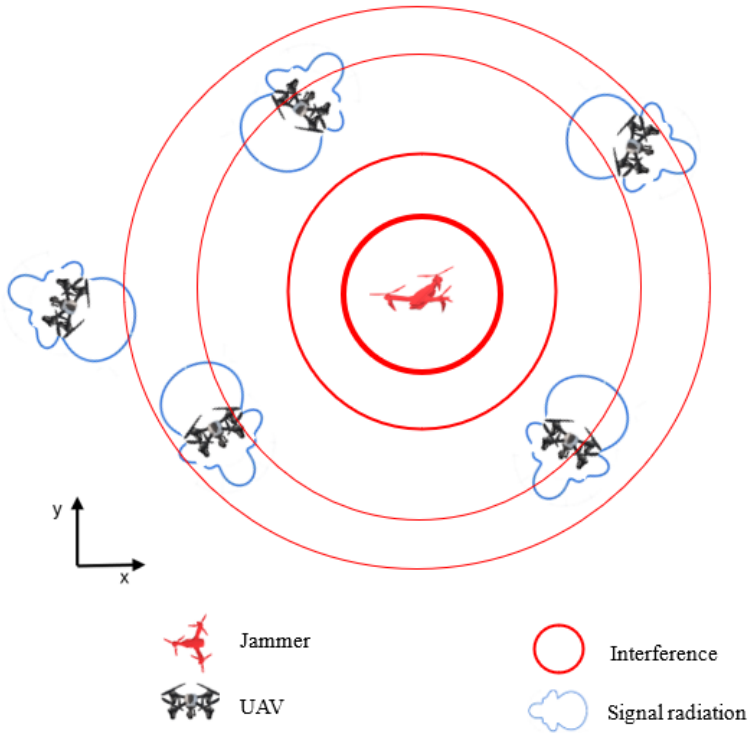


Figure 2: Proposed communication system.

The model studied requires some assumptions, namely:

- **Communications Protocol**

  This simplification assumes that all the coordination among the UAVs is transmitted instantaneously and free from interference.

  This would make sense in a scenario where short swarm coordination messages can be transmitted omni-directionally using a more robust, jammer-resilient, low bitrate transmission mode, while the data itself (e.g., camera images) require higher bitrate modes, making them vulnerable to jamming.

- **Time Discretization**

  Time is assumed to be divided in slots. Optimization is only calculated for the target position that the UAV swarm should occupy at the end of each time slot. Optimization of UAV routes and intermediate positions during the time slot is not considered, being relegated for future work.

In the military environment, assigning areas of interest is a crucial practice that offers various operational, strategic, and security advantages and limits the objectives that can be achieved. This approach ensures the

efficiency and effectiveness of missions, minimizes risks and maximizes coordination between different units and systems.

Thus, during the simulation of the scenarios explained in Chapter 4, defining an area with discrete points where the UAVs could take up their positions was necessary. The discretization of the positions was adjusted with a view to realistic applicability in an operational environment.

In order to reduce the complexity of the simulation model, it was not feasible to exchange the data generated between a wireless network simulator and the RL environment created using OpenAI Gym. Consequently, all the features that characterize a wireless communications model had to be implemented through functions so that the environment created could present adjusted values that were close to what was expected. The main features implemented are listed below.

3.1.1 Communications

As mentioned above, for the purposes of this thesis, UAVs are considered be equipped with smart antennas, able to perform fast reconfiguration of the beam direction. This implies that for each given position, the reception and transmission gains between system nodes are calculated.

In contrast, the jammer was characterized by high-power omnidirectional interference compared to UAVs.

3.1.2 Antenna Modeling

In this thesis, the antenna defined in [37] was adopted. A model was then developed, which provides a discrete radiation pattern, defining the correspondence between each spatial angle and the corresponding antenna gain. Considering that the UAV carries a small phased array antenna (uniform linear array), according to the general theory of the phased array antenna, the pattern of the uniform linear array is described by the Equation 11.

$$E(\theta) = \left| \frac{1}{N} \frac{sin\left[\frac{N}{2}(\frac{2\pi}{\lambda} d_1 sin\theta - \varphi)\right]}{sin\left[\frac{1}{2}(\frac{2\pi}{\lambda} d_1 sin\theta - \varphi)\right]} \right| \tag{11}$$

Where N is the number of elements, d_1 is the spacing of the elements, φ is the phase difference of the elements, λ is the operating wavelength and the corresponding carrier frequency is f .

Since the radiation energy of the phased array antenna is generally concentrated in the main lobe and the first side lobe, in order to simplify the modelling process, the pattern is approximated by the $sinc$ function $y = sin(ux)/ux$.

Based on the work carried out in [38], the typical radiation propagation model of a directional antenna can be represented in Figure 3.

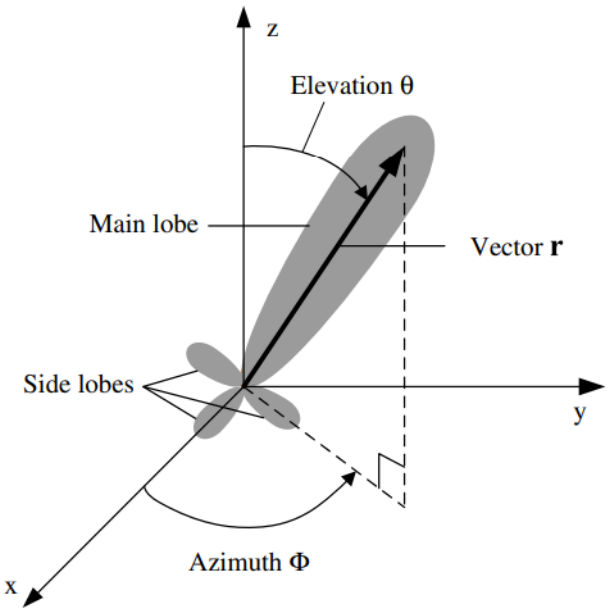


Figure 3: Radiation Pattern of a Directional Antenna [38].

3.1.3 Radiation Pattern

As mentioned above, realistic modelling of the system under investigation requires assigning directional capabilities to each integral node. As such, it was necessary to implement a realistic radiation diagram to correctly and accurately obtain the transmission and reception gains between UAVs and between each UAV and the jammer.

In [39], efficient antennas are studied for radar applications mounted on UAVs, with directional beam characteristics and 360° coverage. In this work, frequencies of 5.9 GHz were used. By analyzing this article, the typical gain values of the directional antenna studied were extracted and applied in this investigation. The radiation diagram used is visible in Figure 4.

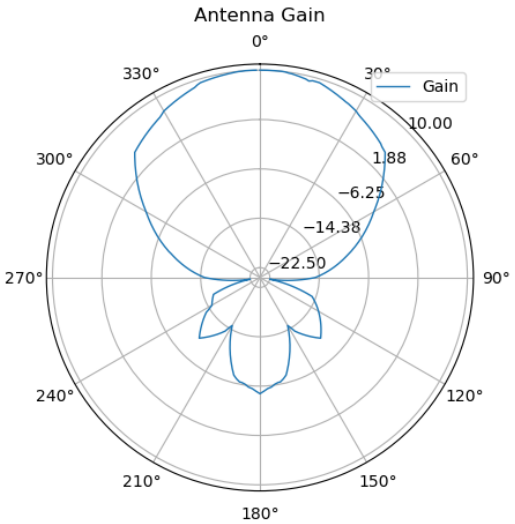


Figure 4: Radiation diagram for the directional antenna.

By evaluating the image, it's easy to distinguish one main lobe with the highest gain, providing the UAVs' directional capacity and three secondary lobes. There are also two lower gain zones, between ninety and one hundred and twenty degrees and then between two hundred and forty and two hundred and seventy degrees. It is, therefore, essential to apply both these angular windows and the radiation lobes to ensure the measures' success.

The model created in this thesis is adapted to receive different radiation patterns and to work with different frequencies.

3.1.4 Positions Formulation

This simulation environment aims to model and optimize the behaviour of a communication network between a swarm of UAVs in a 2D scenario, taking into account the positions of the UAVs and the directions of their antennas. The optimization aims to maximize the average and minimum capacity of the communication links between the UAVs while considering the presence of a jammer trying to interrupt communications.

In this way, an area of interest for the swarm was established where each UAV would have complete freedom of positioning within it. Given the complexity of exploring the environment continuously, the area was discretised into 99 positions, each five meters apart.

Regarding the antenna directions, in each topology evaluated, the antennas can take on any value between 0 and 360 degrees to reduce the interference experienced.

3.1.5 Sharing positions

By Constantly sharing positions, whether through sensors (e.g., radar, cameras, etc.) or communication protocols (e.g., Wi-Fi, Zigbee, LoRa, etc.), each UAV in the swarm can acquire and calculate the positions and directions of transmission and reception between each pair of UAVs. The same happens between the nodes of the communications system and the jammer.

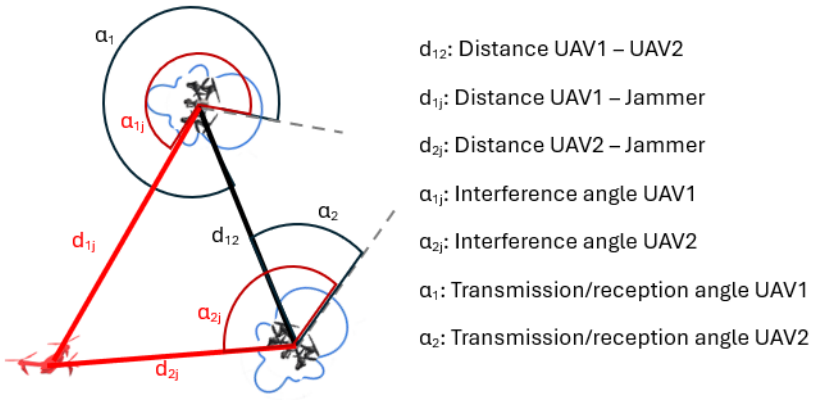


Figure 5: Acquisition of reception/transmission distances and angles.

The losses will be calculated through the distance values and the respective angular values. Based on this, the transmission, reception and interference powers will be determined.

3.1.6 Link capacity

The capacity of any link established between UAVs can be calculated by acquiring the above values.

To do this, a loss value is obtained as a function of distance, following the log-distance path loss model, which can be described using the Equation 12.

$$L = L_0 + 10\gamma \log_{10}\left(\frac{d}{d_0}\right) + X_g \tag{12}$$

Where:

- L_0 is the loss in decibels (dB) at the reference distance d_0.
- γ is the loss exponent.
- d is the value of the link distance.
- d_0 is the reference distance.
- X_g is a normal random variable with zero mean, reflecting the attenuation caused by fading.

This value is then entered into the Equation 13 that returns the value of the power received.

$$P_{\text{RX\_dBm}} = P_{\text{TX\_dBm}} + \text{G}_{\text{TX}} + \text{G}_{\text{RX}} - L \tag{13}$$

Where:

- $P_{\text{RX\_dBm}}$ is the value set for the transmission power of all UAVs.
- G_{TX} is the antenna gain at the respective angular value of the transmitting UAV.
- G_{RX} is the antenna gain in the respective angular value of the receiving UAV.
- L is the loss value found earlier.

Using the same equation, the interference power of the jammer felt by each UAV is found.

Finally, the capacity of the link studied is determined. The power values determined are converted into linear units (Watt) to calculate the SNR value, using the Equation 14.

$$\text{SNR} = \frac{P_{\text{RX}}}{P_{\text{RX\_ruído}}} \tag{14}$$

And the capacity found by Equation 15.

$$C = B_{\text{Hz}} \cdot \log(1 + \text{SNR}) \tag{15}$$

Where B_{Hz} is the bandwidth value used by the communications system.

3.1.7 Network capacity

In the implemented algorithms, the values of the capacities play a crucial role in the process of convergence and optimisation of the agent's policy.

Therefore, it was necessary to represent two metrics essential when evaluating a communications system in the reward value: the average capacity and the minimum ”bottleneck” value.

Each state of the environment has an associated capacity matrix (non-symmetrical matrix) where the capacities of all the UAVs are stored. However, when faced with a malicious agent, it is in the interest of this communications system to know which paths maximise transmission capacity.

In this way, in the system under study, each state can be represented as a graph in which the lowest-cost path between two vertices is to be found. This problem can be solved easily by applying the Dijkstra algorithm.

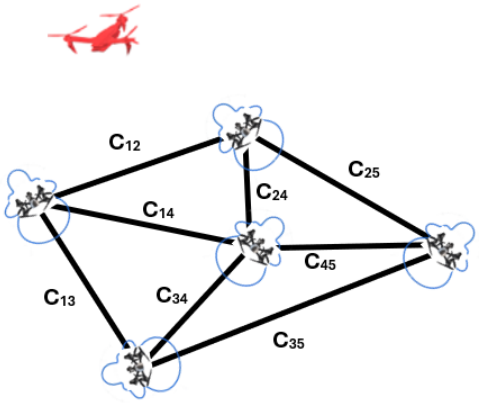


Figure 6: Path cost graph.

Where the weight of each edge of the graph is inversely proportional to the value of the link capacity represented on the edge by Equation 16.

$$\text{Weight} = \frac{1}{C_{link}} \tag{16}$$

By implementing this algorithm in each state, all the lowest-cost paths from one UAV to the others are found. Once the best paths have been determined, the lowest capacity value is taken, which will represent the bottleneck value of our system. This value, together with the average link capacity, will be used for the state being analysed.

The Table 2 shows how the values for each optimised path are structured.

Table 2: Paths and bottlenecks evaluated for each state

Origin	**Destination**	**Path**	**Bottleneck [bps]**
0	1	[0, 1]	14.913
0	2	[0, 1, 2]	14.913
0	3	[0, 3]	8.789
1	0	[1, 3, 0]	6.513
1	2	[1, 2]	537.420
...	...	...	...

3.2 Problem Formulation

This research proposes to develop anti-jamming strategies for swarms of UAVs in the face of the detection of a adversary jammer, optimizing the relative positions of the UAVs and the directions of their individual antennas in order to maximize the average capacity and the minimum capacity of the links established between the UAVs through coordination between the nodes of the system.

By implementing a simulation environment modelled in two-dimensional form, each UAV will assume a position and adjust the direction of its antenna.

Each UAV will be associated with:

- **Position:** Defined in Cartesian coordinates (x, y).
- **Antenna direction:** Antenna orientation angle in the range 0° to 360°.
- **Capacity to communicate with other UAVs:** The ability to transmit data is influenced by the distance between nodes, transmission power, established directions, and jammer interference.

3.2.1 Objective Function (OF)

Using the design variables, position and antenna, the objective function in the formulation of this problem can be described by the equation 17.

$$\text{Maximize} \quad f_{objective}(p, \theta) = C_{avg}(p, \theta)^{\alpha} \cdot C_{min}(p, \theta)^{\beta} \tag{17}$$

where:

- p represents the positions of the UAVs.
- θ represents the directions of the antennas.
- α and β are weights that determine the relative importance of each term in the optimization function.
- C_{avg} is the average capacity of the links established in the network.
- C_{min} is the minimum capacity of the links established in the network.

The value of C_{avg} maximises the sum of the path capacities but allows some UAVs to be disconnected.

The value C_{min} maximises the quality of the link with the worst quality but leads to the remaining links being worse than possible.

A non-linear approximation was used so that, given the different orders of magnitude of the metrics, the average capacity would not take control of the direction taken by the algorithm.

The choice of α and β values depends on the relative importance you want to give to each objective. For a more balanced system, you can start with α = β = 1 and adjust as necessary to meet the specific use requirements.

3.2.2 Adaptable variables

This problem involves establishing numerous variables that can be easily adjusted and adapted, making it possible to test in entirely different environments. This capacity was respected from the outset to adjust and

compare the results, as shown in the following section. In this problem, we can define the variables presented in Table 3.

Table 3: Adaptable variables for UAV communication system

Symbol	Description	Unit
B	Bandwidth available for UAV communication	Hz
P_{jammer}	Transmission power of the jammer	dBm
P_{txUAV}	Transmission power of each UAV	dBm
$P_{txJammer}$	Transmission power of jammer	dBm
N_{UAV}	Number of UAVs in the system	
L	Length of the area of interest	metters
W	Width of the area of interest	metters
S	Spacing between discretised points	metters
R	Radiation pattern of UAV antennas	
γ	Loss exponent	
d_0	Reference distance	meters
T	Time limit for the output of the simulation result	seconds

3.3 Genetic Algorithm Approach

Given the complexity of the environment under study, exploring an action space can become computationally demanding. It has, therefore, become essential to compare results using different approaches.

A GA was, therefore, developed as an optimisation method with the aim of reducing computational complexity and converging on a solution that is close to the optimal solution. This algorithm is inspired by the processes of natural selection and biological evolution.

For a better understanding, it is necessary to emphasise the concept of chromosomes in this context. In the specific context of UAVs and antenna direction, a chromosome is a sequence of values (genes) that code the antenna directions (and subsequently the positions) for each UAV. In this research, each chromosome is made up of a number of genes equal to the number of UAVs under study. Each gene encodes a direction acquired by the UAV. Figure 7 is intended as an example.

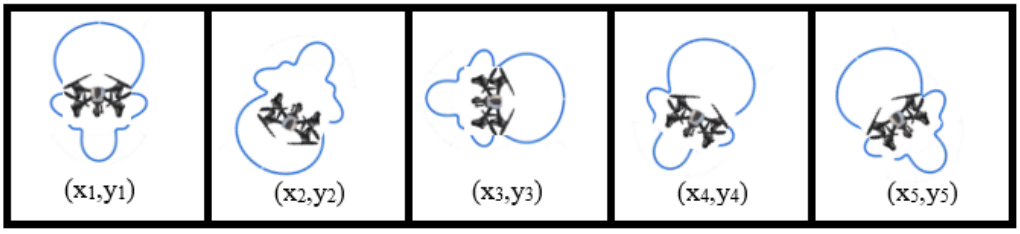


Figure 7: Composition of a chromosome.

This will be the basic structure for the genetic approach.

3.3.1 Initialization of the population

To start the evolutionary process, the algorithm generates an initial population of random chromosomes.

At this stage, a random number of points is selected on the grid equal to the number of drones in the swarm. Next, the UAVs' antennas' directions are randomly generated from 0 to 360 degrees. Finally, the positions and directions are concatenated into a single array (or vector) representing the complete chromosome. The chromosome will have the following structure:

$$[x_1, y_1, x_2, y_2, \ldots, x_n, y_n, \theta_1, \theta_2, \ldots, \theta_n]$$

Where (x_i, y_i) represents the position of UAV i, and θ_i represents the direction of the antenna of UAV i.

3.3.2 Evaluating fitness function

This function will assess the effectiveness of the directions and positions chosen to maximise the objective function. In this way, the fitness function corresponds directly to the OF described in Equation **??**.

At this stage, the positions and directions of each UAV's antennas are extracted and sent directly to the environment as an action. Once the action has been executed, the environment will provide the capabilities matrix in the following format:

$$\begin{bmatrix} 0 & C_{01} & C_{02} & C_{03} \\ C_{10} & 0 & C_{12} & C_{13} \\ C_{20} & C_{21} & 0 & C_{23} \\ C_{30} & C_{31} & C_{32} & 0 \end{bmatrix}$$

The average and minimum capacity are found after applying the Dijkstra algorithm to the calculated matrix. The value of the fitness function is then calculated, which, in this case, will be equal to the objective function.

3.3.3 Genetic Selection

This stage has allowed the swarm to evolve towards a topology that is as favourable as possible for communications. To this end, the following stages were developed:

a) Evaluating fitness

The fitness value in relation to the environment is associated with all the chromosomes in the population. This is the value used to compare individuals.

b) Tournament

At this stage, groups of chromosomes were created, and only the chromosome with the highest fitness value in each group moved on to the next generation.

The *tournament size* variable was set to 5 chromosomes per group. This value reflects the balance between creating a selective pressure that could give prominence to the best chromosomes but occasionally allow some variability without jeopardising a significant time increase in the algorithm's convergence.

The tournament is held until the entire new generation has been selected.

3.3.4 Crossover

The aim of this operator is to explore new areas of the environment by combining characteristics of existing solutions. This allows GA to potentially discover better solutions that inherit the qualities of their parents.

The crossover rate was set to occur randomly at 20% of the population. A single crossover point was defined for each pair of chromosomes. However, this point was defined randomly over the length of the chromosome. The two children are then created by combining the parts of the parents, as follows:

- **Child1**

  The first part of parent1 (before the crossover point) is combined with the second part of parent2 (after the crossover point).

- **Child2**

  The first part of parent2 is combined with the second part of parent1.

Figure 8 demonstrates the crossover process described.

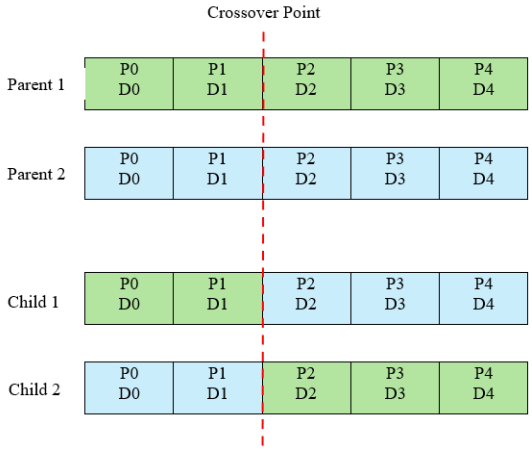


Figure 8: Crossover process.

After the crossover, it was found that some of the chromosomes generated had genes with repeated positions. A check was therefore carried out on all the chromosomes. If there were genes with the same position, the repeated gene with the highest index would occupy the nearest free position in the grid.

3.3.5 Mutation

This operator helps maintain genetic diversity in the population and prevents the algorithm from getting stuck in local maxima or minima.

For each UAV, it is decided whether or not its position should be mutated based on a defined *mutation rate*. If there is a mutation, the UAV is assigned a new random position, available in one of the eight closest positions on the grid, and the direction of its antenna undergoes a random deviation in the range [-20, 20] degrees. This process only aims to make small changes but always maintains stability in the algorithm's evolution. To this end, the *mutation rate* was set to a range between 5% and 15%.

Figure 9 shows the mutation process.

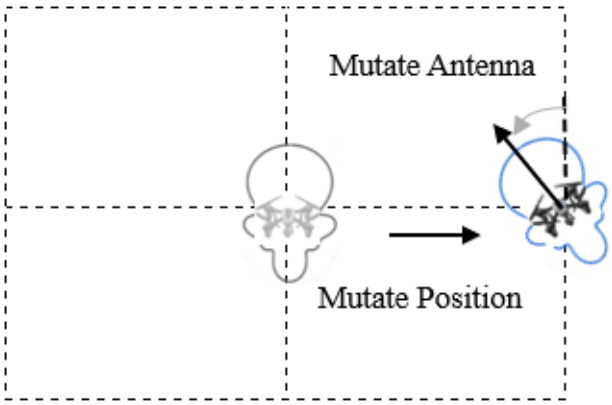


Figure 9: Mutation process.

3.3.6 Implementation sequence followed

The process begins with initializing the genetic algorithm's parameters, such as the population size, mutation rate, crossover rate and maximum number of generations.

Next, the chromosomes are encoded, where each chromosome represents a possible solution. After this, the population is initialized, randomly generating potential solutions.

The fitness value is calculated for each chromosome in this population. The goal of the algorithm is to maximize the fitness function.

After this evaluation, it checks whether the number of current generations has already reached the maximum number of generations defined. If there are still generations to be processed, the algorithm continues. At this point, genetic selection occurs, where the best chromosomes are more likely to be selected to generate new ones.

From this selection, the crossover and mutation operators are applied. Crossover combines two solutions to create new ones, while mutation introduces small random changes to the chromosomes, ensuring diversity in the population.

These operations generate a new population, and the process begins again. The fitness evaluation, selection, crossover and mutation cycle is repeated until the maximum number of generations is reached.

The sequence of the implemented functions can be analysed from Figure 10.

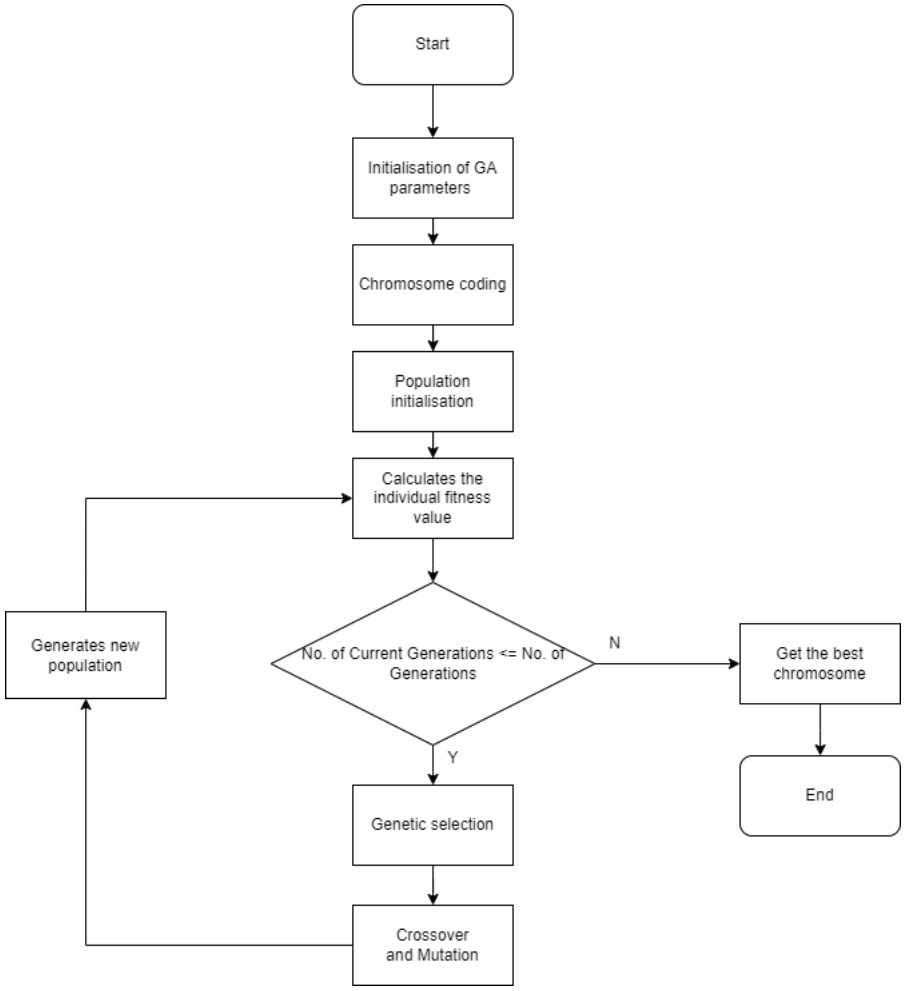


Figure 10: Flowchart of the GA used.

3.3.7 Encapsulation

Optimization using GA required reducing the complexity of the action space. Given the structure of each chromosome, made up of established positions and directions, it was possible to divide the optimization process into two parts:

- **Phase One: Position Optimization**

  In the first phase of GA, the positions of the UAVs are optimized. The focus is finding the best arrangement of UAVs in space to maximize the network's connectivity and communication capacity.

  This phase aims to distribute the UAVs efficiently, ensuring that the distances between them allow for good communication, minimizing interference and maximizing range.

  After this phase, the positions of the UAVs are fixed, and the next step is to adjust the directions of the antennas to optimize communication further.

- **Phase Two: Optimizing Antenna Directions**

  In the second phase, GA focuses on optimizing the directions of the UAVs' antennas. Here, the positions are already fixed based on the results of the previous phase.

This phase aims to adjust the antenna angles to improve signal gain and reduce interference, taking into account the optimized positions of the UAVs.

Thus, GA seeks to maximize communication capacity by finely adjusting the antenna directions to ensure that each UAV can communicate efficiently with the others.

Separating the optimization into two phases reduces the complexity of the GA search. If GA tried to optimize positions and directions simultaneously, the search space would be much larger and more challenging to explore.

Finding the best UAV positions first ensures the antenna optimization phase has a solid foundation. The antennas will only be adjusted once the best positions have been found, which avoids unnecessary adjustments to sub-optimal positions.

In this way, GA is more likely to converge on a better solution in each phase.

3.4 RL approach

This section discusses the application of RL to solve the proposed optimisation problem. RL is a powerful technique in which an agent learns to make decisions based on interactions with a dynamic environment, adjusting actions to maximise accumulated rewards. In this context, this approach allows the system to dynamically adapt to variables such as the positions of the UAVs and the directions of their antennas to optimise the network's communication capacity.

3.4.1 Markov Decision Process (MDP) formulation

The formulation of the MDP meant that all the characteristics found in the environment had to be structured so that the agent could establish contact, implement actions and receive results from the states generated in the topology of the UAV swarm.

It was, therefore formulated as follows:

a) Set of States

The state fully describes the current condition of the environment. Each state $s \in S$ contains all the information necessary for the agent to know what is happening in the environment.

Each state s contains:

- **Positions**

  The current positions of the UAVs in a two-dimensional environment.

  For n UAVs, the Positions vector is made up of $2n$ positions, as follows:

$$Positions = [x_1, y_1, x_2, y_2, ..., x_n, y_n]$$

- **Directions**

  The directions of the antennas of each UAV, orientated in a direction θ between 0 and 360 degrees.

  For n UAVs, the Directions vector is made up of n values, as follows:

$$Directions = [\theta_1, \theta_2, \theta_3, ..., \theta_n]$$

- **Jammer position**

  The position of the jammer can be represented by its coordinates (x_j, y_j).

$$Jammer = [x_j, y_j]$$

- **Capacity Matrix**

  The matrix of capacities between the UAVs is an nn matrix, where each entry C_ij represents the communication capability between UAVs i and j.

  This matrix is represented as a vector of size n^2.

$$Capacities = [C_{12}, C_{13}, ..., C_{n-1,n}]$$

This way, the state vector is obtained by concatenating all the previous vectors. For a number n of UAVs in the swarm, the vector s has a size of $2n + n + n^2 + 2$.

b) Set of Actions

Actions are the choices the agent can make in each state. For each state $s \in S$, the agent can select an action $a \in A(s)$ that influences the transition to the next state.

Actions include:

- **x and y movements on each UAV**

  UAVs can move in two-dimensional space by changing their x and y coordinates.

  Thus, for each action a taken, a vector composed of the x and y displacements of each UAV is generated, of size $2n$.

$$Displacement = [dx_1, dy_1, dx_2, dy_2, ..., dx_n, dy_n]$$

- **Adjusting antenna directions**

  The agent can change the direction angle of the antennas between 0 and 360 degrees.

  The deviation of the directions from the directions is stored in the following vector:

$$Deviation = [d\theta_1, d\theta_2, ..., d\theta_n]$$

A complete action vector a for n UAVs is made up of $3n$ elements:

$$a = [dx_1, dy_1, d\theta_1, dx_2, dy_2, d\theta_2, ..., dx_n, dy_n, d\theta_n]$$

c) State Transition Function

Transition function will describe how the positions of the UAVs change according to the chosen movements, how the directions of the antennas affect the communication capacity and how the jammer interference conditions and the relative position of the UAVs change the state of the communication network after each action taken by the agent.

New State s'

The new state s will be the result of applying action a to state s, reflecting the following changes:

- **Updating Positions**

  Each UAV will have a new position based on its displacement dx_i and dy_i, given by the Equation 18 and Equation 19.

$$x_i' = x_i + dx_i \tag{18}$$

$$y_i' = y_i + dy_i \tag{19}$$

  The new positions must respect the network boundaries defined in the environment.

- **Updating Directions**

  The direction of each antenna will be adjusted according to the action taken.

$$\theta_i' = \theta_i + d\theta_i \tag{20}$$

- **Communication Capacities**

  The capacity of the communication links between the UAVs will be recalculated based on the new antenna positions and directions. The function f for calculating the capacity of a link between two UAVs i and j involve the distance between them, the angular values of the antennas and the transmit and receive powers.

$$C_{ij}' = f(d_{ij}', \theta_i', \theta_j', P_{tx}, P_{jammer})$$

- **Transition Function**

  The transition function $P(s|s, a)$ can be described as a mapping that defines how the state s changes to s after the execution of the action a. The transition function is described by Equation 21.

$$P(s' \mid s, a) = \begin{cases} x_i' = x_i + dx_i \\ y_i' = y_i + dy_i \\ \theta_i' = \theta_i + d\theta_i \\ C_{ij}' = f(d_{ij}', \theta_i', \theta_j', Ptx, P_{jammer}) \end{cases} \tag{21}$$

d) Reward Function

The reward function $R(s, a)$ will be calculated based on:

- **Average Capacity**

  This term encourages drones to adjust their antenna positions and directions to maximize the overall communication capacity in the network. Greater communication capacity results in a greater reward.

- **Minimum Capacity**

  This term also ensures that the weakest link in the communication is considered. This is important to ensure that there are no network partitions.

The equation can then be described identically to the OF, described by the equation 17.

f) Policy

The policy defines how the agent adjusts the UAVs' positions and the antennas' directions to optimise communication. The policy will learn to select actions dynamically, depending on the interference conditions and the communication capacity between the UAVs.

Policy improvement uses the Equation 5 to determine a new policy π' that chooses actions that maximise the expected reward.

g) Goal

The goal of MDP is to find an optimal policy π^* that maximises the expected value of the rewards accumulated over time.

The agent learns to adjust the positions and directions of the UAVs' antennas optimally to maximise the performance of the communication network. The Agent-Environment interaction can be described by Figure 11.

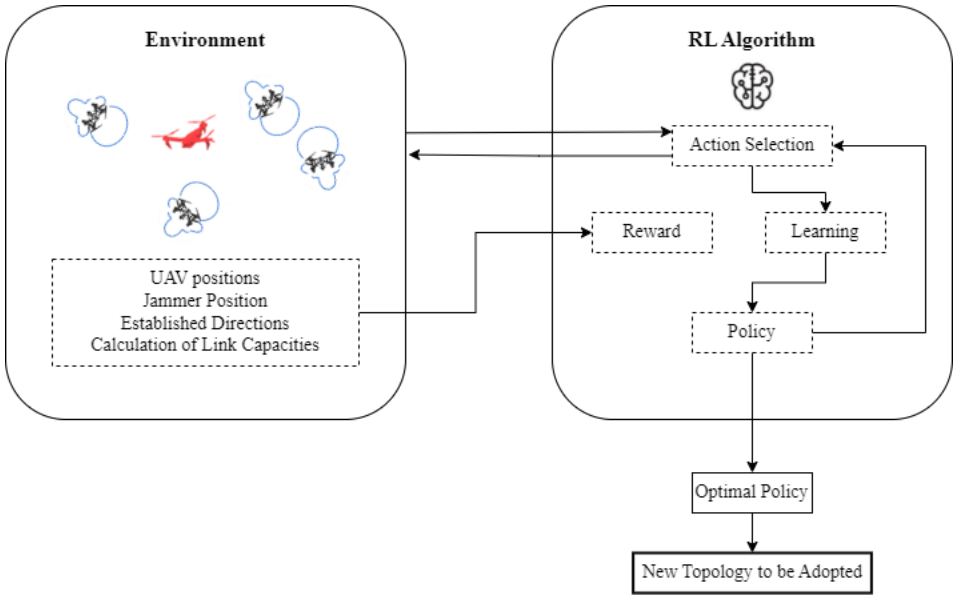


Figure 11: Agent-Environment Interaction, adapted from [40].

3.4.2 *Proximal Policy Optimization* (PPO)

Several characteristics led to the choice of this algorithm: providing a good balance between simplicity and robustness, reusing data samples several times, which means it can learn with fewer interactions with the environment, and its flexibility.

The implementation of this algorithm was achieved through different parts:

Callback Function

The callback in the code serves to monitor and record information during the training of the PPO model. It was implemented to precisely track the rewards obtained during the agent's training at regular time intervals.

Once the validation frequency has been set, the reward results of the last few episodes are displayed. This analysis allows the algorithm's training progress to be visualized so that it can be parameterized to suit the algorithm's behaviour.

Importing Data Set

A pre-train was implemented with a data set to reduce complexity and achieve better results. In Comma-Separated Values (CSV) format, the file was imported containing various values, such as the jammer's position, the UAV's position, the reward, etc.

The CSV file was created by generating random scenarios where, for each scenario, only the five topologies with the best reward values were selected.

Pre-Training

The PPO algorithm is implemented using the stable-baselines library3. PPO adjusts the agent's policy via the OF and limits the magnitude of updates to the policy, ensuring that changes are gradual via the Clip Objective Function.

Using the results obtained through Callback, the number of steps, number of epochs, batch size, learning rate, clip range, and entropy coefficient parameters were adjusted.

The Algorithm 1 shows how the pre-train was carried out.

Algorithm 1 Pseudo-Code of PPO Pre-Training

function TRAIN_PPO(env, total_timesteps, learning_rate, n_steps, batch_size, n_epochs, clip_range)
    Create an instance of the vectorised environment
    Initialise the policy $\pi(\theta)$ and the value $V(\theta)$ with weights θ
    for $t = 0$ to total_timesteps step n_steps **do**
        Obtain data on the agent's interaction with the environment (states, actions, rewards, next_states)
        Estimate the advantages ($\hat{A}_t$)
        Store the acquired data in batches of size batch_size
        for epoch = 1 to n_epochs **do**
            for each batch of data collected from the environment of size batch_size **do**
                Calculate the probability ratio $r_t(\theta)$
                Calculate the value of *loss* using $L^{CLIP}(\theta)$
                Calculate the total loss: total_loss = policy_loss + c_1 * value_loss
                Compute the gradient of the total loss with respect to parameters θ
            end for
        end for
    end for
    Save the trained model
    Return the trained model
end function

Prediction

Once the model has been trained and saved, the predict function is used to determine the best configuration predicted by the algorithm. Given the speed of the prediction generated by the algorithm, a cycle is created with several iterations to be defined, large enough for redundancy of results and small enough not to jeopardize the simulation time, where the predicted topologies and their respective reward value are stored.

At the end of the cycle, the topology with the best reward value is rendered.

Algorithm 2 Pseudo-Code of the Prediction Loop

Input: $num\_test\_episodes$
Initialize $best\_reward \leftarrow -\infty$, $best\_episode \leftarrow 0$, $best\_episode\_steps \leftarrow []$
for $episode$ from 1 to $num\_test\_episodes$ **do**
    $state, \_ \leftarrow env.reset()$
    $episode\_reward \leftarrow 0$
    $episode\_steps \leftarrow []$
    while not $done$ **do**
        $action \leftarrow agent.select\_action(state)$
        $next\_state, reward, done, info \leftarrow env.step(action)$
        $episode\_steps.append((state, action))$
        $state \leftarrow next\_state$
        $episode\_reward \leftarrow episode\_reward + reward$
    end while
    if $episode\_reward > best\_reward$ **then**
        $best\_reward \leftarrow episode\_reward$
        $best\_episode \leftarrow episode$
        $best\_episode\_steps \leftarrow episode\_steps$
    end if
end for
Output: $best\_episode, best\_reward$

Running the Algorithm

The PPO algorithm implemented followed the sequence shown in Figure 12.

3.4.3 *Soft Actor-Critic* (SAC)

SAC is a popular algorithm in RL, *off-policy*, meaning it can learn from data not generated by the current policy. This feature allows it to efficiently reuse past experiences, which is helpful in complex and continuous environments such as this, where fine adjustments to UAV positions and antennas can significantly impact the quality of communication.

Consequently, the structure of the algorithm was created:

Actor Network

The Actor Network decides how the UAVs should move or adjust the antennas to maximise communication capacity and minimise interference. Entropy will ensure that these decisions maintain a level of randomness to encourage the exploration of new configurations.

The Actor network parameters were adjusted taking into account:

- **Number of layers:**

  Four hidden layers allowed the algorithm to acquire sufficient depth to capture the complexity of the environment, allowing the actor to learn effective policies.

  Two output layers (Mu and Log Std). One output learns the average of the actions (Mu). In contrast, the other output learns the standard deviation (Log Std), allowing the model to capture the uncertainty of the actions, which was crucial to ensure continuous and efficient exploration.

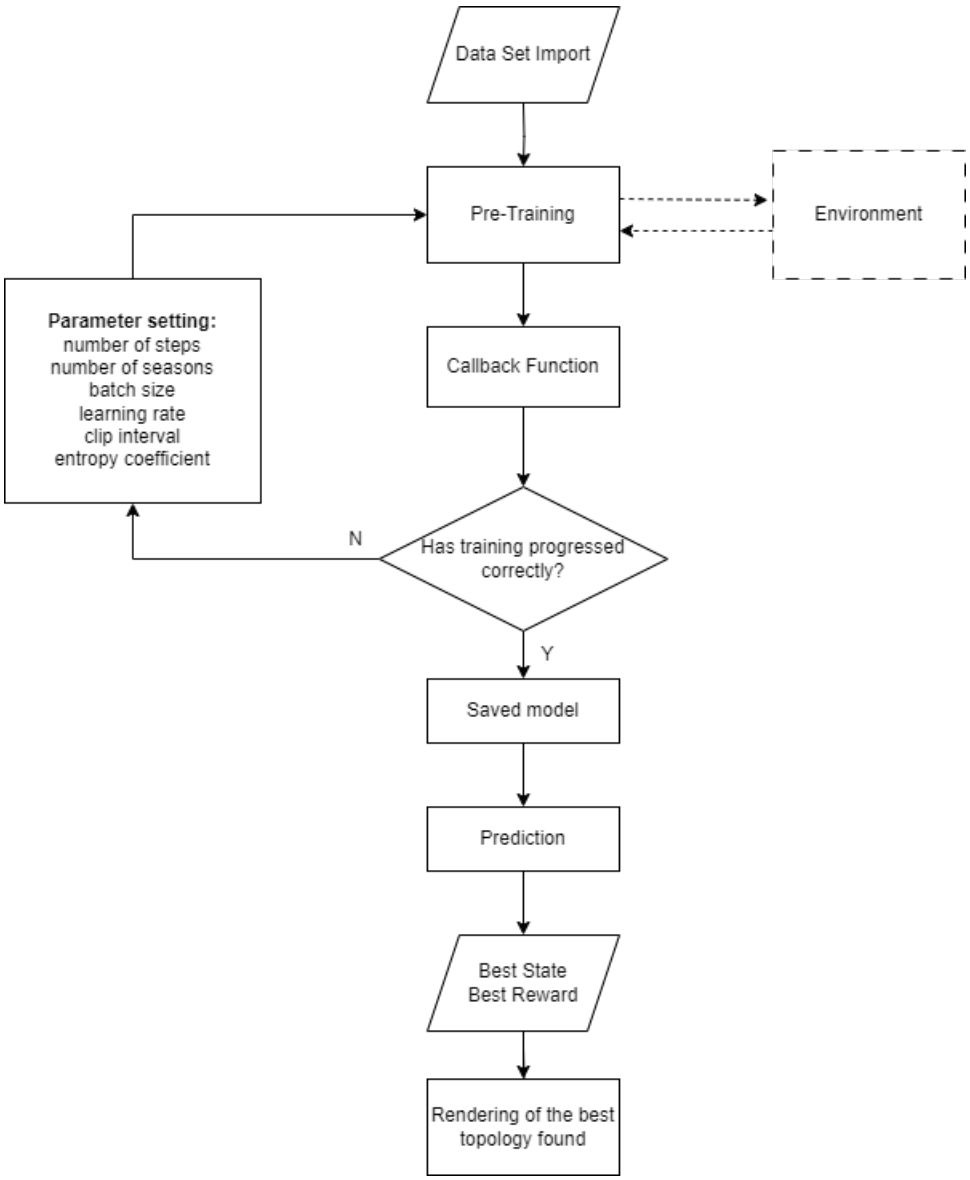


Figure 12: Sequence followed by the PPO algorithm.

- **Layer Dimensions:**

  The input layer has dimensions equal to the observation space and is connected to 512 neurons. The choice of 512 neurons in the first layer provided sufficient initial modelling capacity to process the inputs and extract the first relevant characteristics of the system state.

  The intermediate layers follow the first layer with 512 neurons. This value provides stability and avoids premature reductions in dimensionality. With 512 neurons, each layer was expressive enough to represent the more complex characteristics of the state and actions.

  The number of outputs had to correspond to the number of agent actions, i.e. the number of variables describing changes in the UAVs' movements and antennas.

  The output layer (Mu) generates the actions' average, continuous values corresponding to the UAVs' position and direction adjustments. The output layer (Log Std) generates the logarithm of the standard deviation of the actions, which is essential to allow the SAC to maintain the variability of the actions.

- **Activation Function**

  Rectified Linear Unit (ReLU) was chosen for the hidden layers because it proved computationally efficient and facilitated convergence during training.

  The hyperbolic tangent (Tanh) function in the output layer allowed the actions not to be excessively large, which could destabilize the system.

- **Optimizer**

  The Adam optimizer was chosen for its ability to dynamically adjust the learning rate based on the gradients' first and second-order moments, allowing for a more stable convergence.

The parameters applied to the Actor network can be analyzed in Table 4.

Table 4: Structural parameters of the *Actor* in the SAC algorithm

Parameter	**Actor**
Number of Layers	6 (4 hidden + 2 output)
Layer 1 dimensions	Input dimensions x 512
Layer 2 dimensions	512 x 512
Layer 3 dimensions	512 x 512
Layer 4 dimensions	512 x 512
Output Layer (Mu)	512 x Output Dimension
Output Layer (Log Std)	512 x Output Dimension
Activation Function	ReLU in the hidden layers, Tanh in the action output
Optimiser	Adam

Critic Networks

The Critic1 and Critic2 networks estimate the Q-value of an action-state combination. Represents the expected value of a given configuration of antenna positions and directions and how this configuration affects the quality of communication.

The use of two critical networks that are both trained independently reduced the positive bias that occurs in learning methods, where only one network can overestimate the value of an action. The presence of two critics improves the stability of learning.

The two Critic Networks were implemented according to the following structure:

- **Number of Layers**

  Each critical network has five layers (4 hidden layers + 1 output layer), which offer sufficient depth to capture the complex relationships between the state of the environment, the actions taken by the agent, and the impact of these actions on the quality of communication (the Q-value).

- **Layer Dimensions**

  The first layer receives the concatenation of state and action as input.

  The intermediate layers each have 512 neurons and process the intermediate representation of the state-action.

  The output layer generates the Q value, which is a single value that represents the quality of the specific combination of state and action.

- **Activation Function**

  The ReLU activation function was used in all hidden layers.

- **Otimizer**

  The Adam optimizer was used for both critics due to its robustness and adaptability.

The parameters applied to the Actor network can be analyzed in Table 5.

Table 5: Structural parameters of *Critic 1* and *Critic 2* in the SAC algorithm

Parameter	**Critic 1**	**Critic 2**
Number of Layers	5 (4 hidden + 1 exit)	5 (4 hidden + 1 exit)
Layer 1 dimensions	(State dimension + action) x 512	(State dimension + action) x 512
Layer 2 dimensions	512 x 512	512 x 512
Layer 3 dimensions	512 x 512	512 x 512
Layer Dimensions 4	512 x 512	512 x 512
Output Layer (Q-value)	512 x 1	512 x 1
Activation Function	ReLU on all hidden layers	ReLU on all hidden layers
Optimiser	Adam	Adam

Networks Target

The Target Critic1 and Target Critic2 networks are smooth copies of the main Critic networks and help calculate the target for training the Critic networks.

Thus, Target Critic1 and Target Critic2 were used to calculate the expected future value of the configuration of antenna positions and directions, providing a more stable estimate of the impact of actions on the communication system.

Replay Buffer

SAC uses a replay buffer to store the agent's previous interactions with the environment. Each experience consists of a tuple:

$$(state, action, reward, nextstate, achievement)$$

During the update process, the agent randomly samples a set of past experiences (mini-batches) from the replay buffer to update its neural networks (critic and actor).

The replay buffer was implemented using a double-ended queue, which has the advantage of being limited and automatically discards the oldest experiments when the maximum capacity is reached.

Implementation

The implementation of the SAC can be seen in the Algorithm 3 and 4:

Algorithm 3 Algorithm SAC (Part 1)

function TRAIN_SAC(env, num_episodes, batch_size, gamma, tau, alpha, learning_rate)
Initialise the networks *Critic1*, *Critic2*, *Target Critic1*, *Target Critic2*, and *Actor* with random weights
Initialises the *buffer replay* D
Define the hyperparameters γ, τ, α, and *learning_rate*
for episode = 1 **to** num_episodes **do**
Restarts the environment and gets the initial state s_0
Sets the episode reward with $R = 0$
while not finished **do**
Selects action a_t based on policy $\pi_\theta(a_t|s_t)$ derived from network *Actor*
Executes action a_t in the environment, observes reward r_t and next state s_{t+1}
Saves the transition (s_t, a_t, r_t, s_{t+1}) in the replay buffer D
$R \leftarrow R + r_t$
if D have enough samples **then**
Sample a minibatch of transitions (s_i, a_i, r_i, s_{i+1}) from *buffer replay* D
Update Critical Networks:
Calculates the Q-value target:

$$y_i = r_i + \gamma \min(\text{Target Critic1}(s_{i+1}, a_{i+1}), \text{Target Critic2}(s_{i+1}, a_{i+1})) - \alpha \log \pi_\theta(a_{i+1}|s_{i+1})$$

Updates the critical networks *Critic1* and *Critic2*
Update Actor Network:
Calculates network loss *Actor*:

$$\text{Loss Actor} = \frac{1}{N} \sum_{i=1}^{N} \alpha \log \pi_\theta(a_i|s_i) - \min(\text{Critic1}(s_i, a_i), \text{Critic2}(s_i, a_i))$$

Updates the network parameters *Actor*
end if
end while
end for
end function

Algorithm 4 Algorithm SAC (Part 2)

function TRAIN_SAC(env, num_episodes, batch_size, gamma, tau, alpha, learning_rate)

Update the Target Networks:

Updates the parameters of the target networks *Target Critic1* e *Target Critic2*:

$$\text{Target Critic}_i \leftarrow \tau \times \text{Critic}_i + (1 - \tau) \times \text{Target Critic}_i \quad \text{para } i = 1, 2$$

Save the trained model

Returns the trained model

end function

Chapter 4

Test Scenarios, Performance Evaluation and Results Analysis

This chapter specifies the scenarios in which the proposed model will be tested. Therefore, three test scenarios are proposed, differing in complexity and capability requirements.

This chapter presents the results obtained for each scenario proposed above and discusses their preponderance about the proposed objectives.

Creating realistic scenarios of varying complexity makes it possible to explore behaviour and check performance at different demand levels.

4.1 Static Scenario with Exploitation of Antenna Directionality

In this scenario, the swarm of UAVs remains static, with each node keeping its position static and with no movement over time. The performance evaluation focuses exclusively on the UAVs' ability to direct their antennas so that links can be established without interference from the malicious agent. Figure 13 shows the capabilities of the scenario.

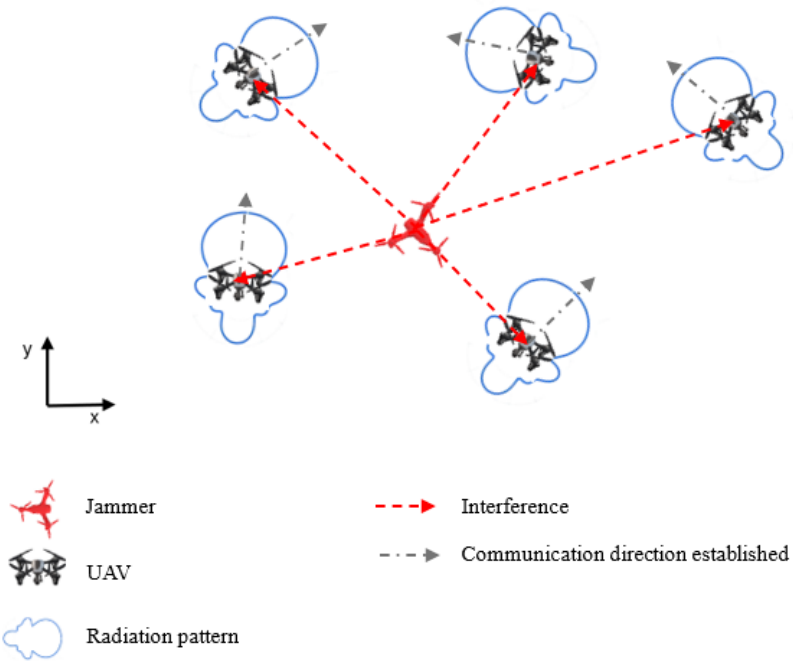


Figure 13: Static scenario exploring antenna directionality.

Evaluation Factors:

- Ability to direct and focus the antennas of each node required for transmission.
- Efficiency in transmitting and receiving signals in a static environment.
- Ability to mitigate interference and optimise spectrum.

4.2 Static Scenario with Exploitation of Antenna Directionality and Positions

In this scenario, in addition to the directionality of the antennas, the spatial positions of each UAV in the swarm are taken into account. The UAVs can adjust their positions about each other while maintaining the directionality of the antennas. In this scenario, the spatial stationarity of the swarm is maintained, with only an individualized movement at the level of each UAV to adjust the topology.

The swarm is assigned a discretised area where each point is a position that each UAV can occupy.

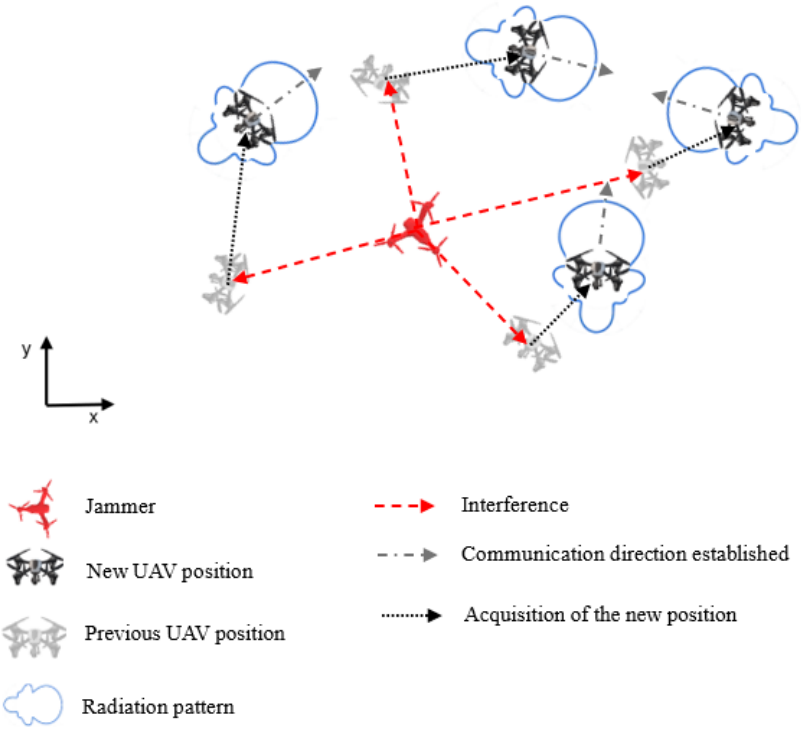


Figure 14: Scenario with exploitation of antenna directionality and UAV positions.

Evaluation Factors:

- Cooperation and coordination between UAVs to optimize the swarm topology.
- Effect of swarm topology changes on system communications.
- Performance in terms of range and connectivity in a dynamic environment.
- Resilience to changes made to the network topology.

4.3 Scenario with a Progressing Swarm

In this scenario, the swarm of UAVs is in constant motion, progressing in space over time. Likewise, the area where the swarm will be free to adopt new topologies will shift at the same speed as the swarm.

The evaluations take into account the swarm's spatial dynamics, requiring continuous adaptation of the antenna's directionality to maintain connectivity.

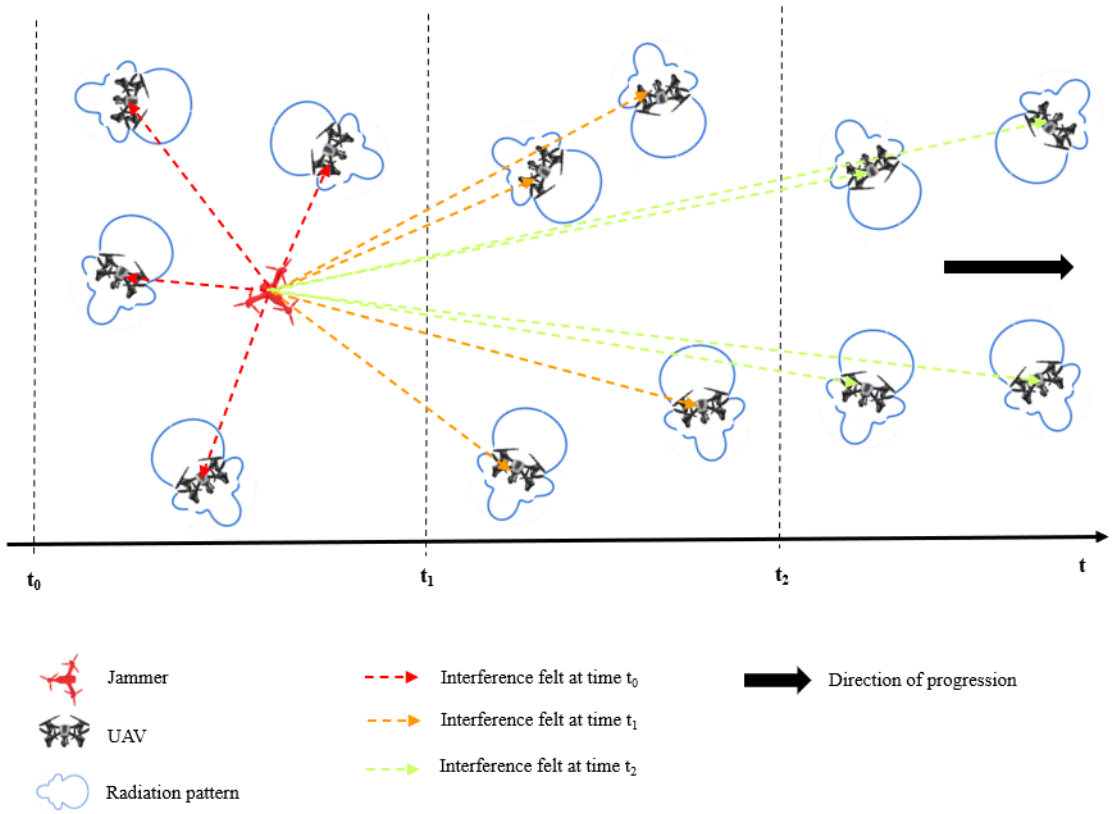


Figure 15: Scenario with swarm in progression.

Evaluation Factors:

To the evaluation factors mentioned in the previous scenario, the following are added:

- Ability to carry out efficient route adaptation during movement.
- Maintaining connectivity in a dynamic environment.

4.4 Parameterisation

As this research proposes, analysing the use of RL in realistic scenarios requires realistic parameterisation. Each parameter controls a specific aspect of the swarm's behaviour and capabilities, directly affecting its performance, connectivity, energy consumption and coverage.

The parameters applied are detailed in Table 6.

Table 6: Parameters applied to the UAV swarm

Parameter	Valor
Number of UAVs	4
Bandwidth	2.4e$^9 Hz$
UAV transmission power	20 dBm
Reference distance d_0	1 m
Loss value for reference distance	30 dB
Interference power of the jammer	100 dBm
Loss exponent γ	2
Size of the area of interest	50m x 40m
Resolution of the network points in the area of interest	5

4.5 Scenario 1: Static Scenario with Exploitation of Antenna Directionality

In this scenario, the effectiveness of the applicability of antenna directionality in a swarm of UAVs where each position remains fixed was studied. In this way, the adaptation of the antenna directions for each jammer position is evaluated.

4.5.1 Result obtained using GA

Initially, the parameters that favoured the solution found in terms of execution time were studied. As such, the following GA parameters were analysed: *population size*, *generations*, *mutation rate* and *crossover* rate. The results can be analysed in Table 7.

Table 7: Study of GA parameterisation

Population Size	Number of Generations	Mutation Rate	Crossover Rate	Objective Function Value	Simulation Time[s]
10	20	0.15	0.8	15810	7
10	20	0.15	0.9	31218	8
10	50	0.15	0.8	38490	10
10	100	0.15	0.8	72333	16
50	10	0.15	0.9	135466	10
50	50	0.15	0.9	197841	13
50	100	0.15	0.9	329988	41
50	200	0.15	0.9	364068	73
100	10	0.15	0.9	204256	14
100	50	0.15	0.9	394567	27
100	100	0.15	0.9	408590	64
100	200	0.15	0.9	402333	143

An analysis of the table shows that the value of the objective function has stabilised. Therefore, it must be realised that increasing the number of *generations* too much will only cause the GA to lose efficiency by increasing the simulation time. In a realistic scenario, the speed with which the new swarm topology is acquired is directly related to the success of the operation it performs.

Table 8 shows the parameters applied to GA.

Table 8: Parameters used in GA

Parameter Value	Value
Population Size	100
Generations	50
Mutation rate	15%
Crossover rate	90%

Figure 16 shows the evolution of the OF value for the directions of the directional antennas and the constant value for an omnidirectional antenna configuration.

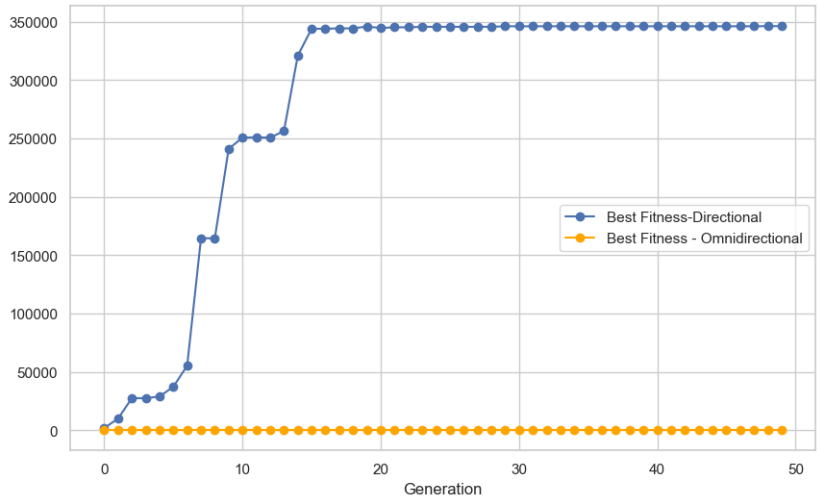


Figure 16: Convergence of objective function values with GA.

The figure shows that the application of directional antennas is indispensable for the communications system studied, with the best fitness value for the omnidirectional configuration being 0.11 in contrast to the value for the directional configuration, which shows a value of approximately 350000.

Once executed, the result shown in Figure 17 was obtained: a simulation time of 29.3s, an average capacity of 2856bps and a bottleneck capacity of 129bps. The figure also shows the direction of the antenna taken by each UAV. For the omnidirectional configuration, an average capacity of 0.89 bps and a bottleneck capacity of 0.12 bps were obtained.

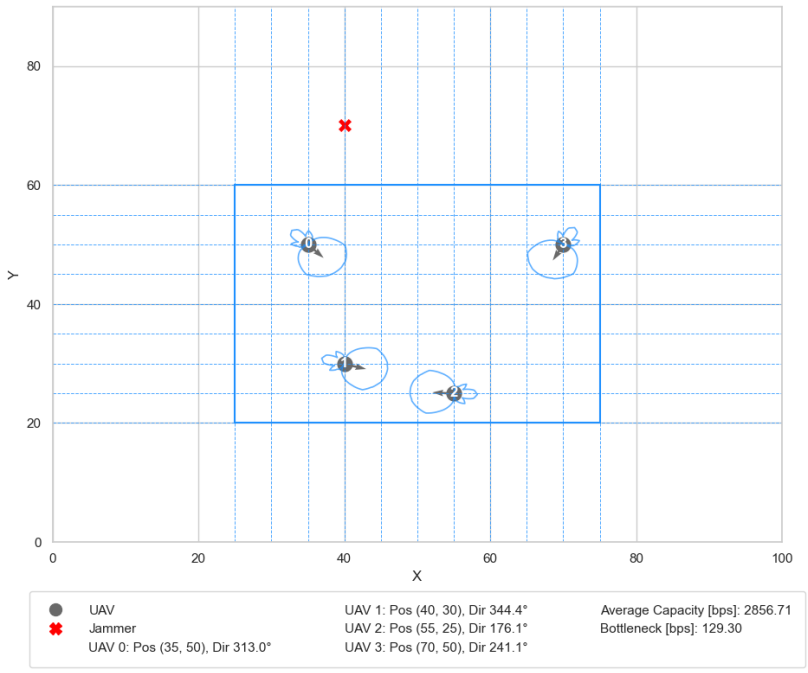


Figure 17: Directions established through the use of GA.

A quick analysis reveals that the antennas favour a direction that avoids interference from the jammer.

Let's look at the behaviour of the two UAVs closest to the jammer, UAV 0 and UAV3. They keep in contact with the jammer in the direction with the lowest gain in the radiation pattern to minimise the interference caused.

In contrast, the two UAVs furthest away prefer to establish direct communication to maximise the capacity between them.

4.5.2 Result obtained using PPO algorithm

Once an adjusted result had been obtained using GA, the PPO algorithm was implemented to try and get an identical result but at a lower cost in terms of execution time.

The behaviour of the reward value curve was then studied over 10000 episodes of training the agent on the environment.

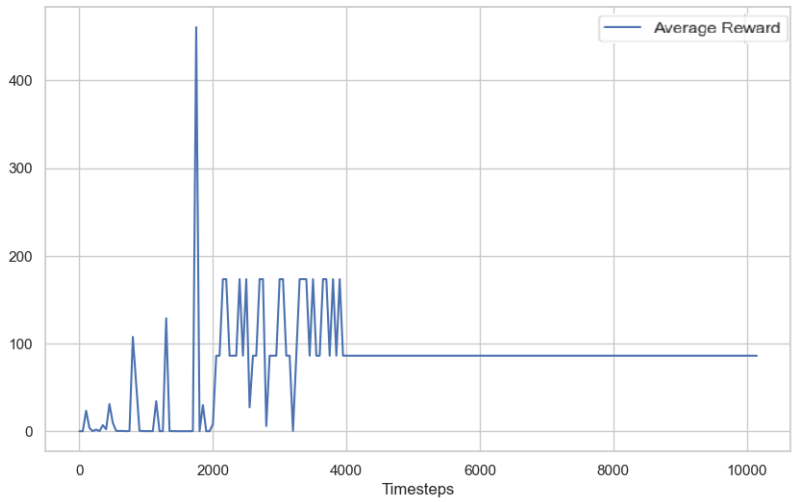


Figure 18: Evolution of the average reward value during training of the PPO algorithm.

The analysis in Figure 18 shows non-ideal behaviour when it comes to finding a promising policy:

There is a relatively high average reward peak around 2000 *timesteps*, which may indicate that the agent found a very good policy by chance or an atypical variation in the training data.

Between 2000 and 4000 *timesteps*, there is significant variability in the average rewards. This fluctuation suggests that the agent is in a period of exploration, trying out different actions and strategies to find out which ones provide the best rewards.

After 4000 *timesteps*, the average reward stabilises around a fairly low value. This behaviour indicates that the agent has found a relatively stable policy but suggests that the agent has converged on a local minimum. He may be stuck with a policy that provides consistent rewards but far from the maximum possible.

Given this policy, the best result predicted by the algorithm is given by the Figure 19.

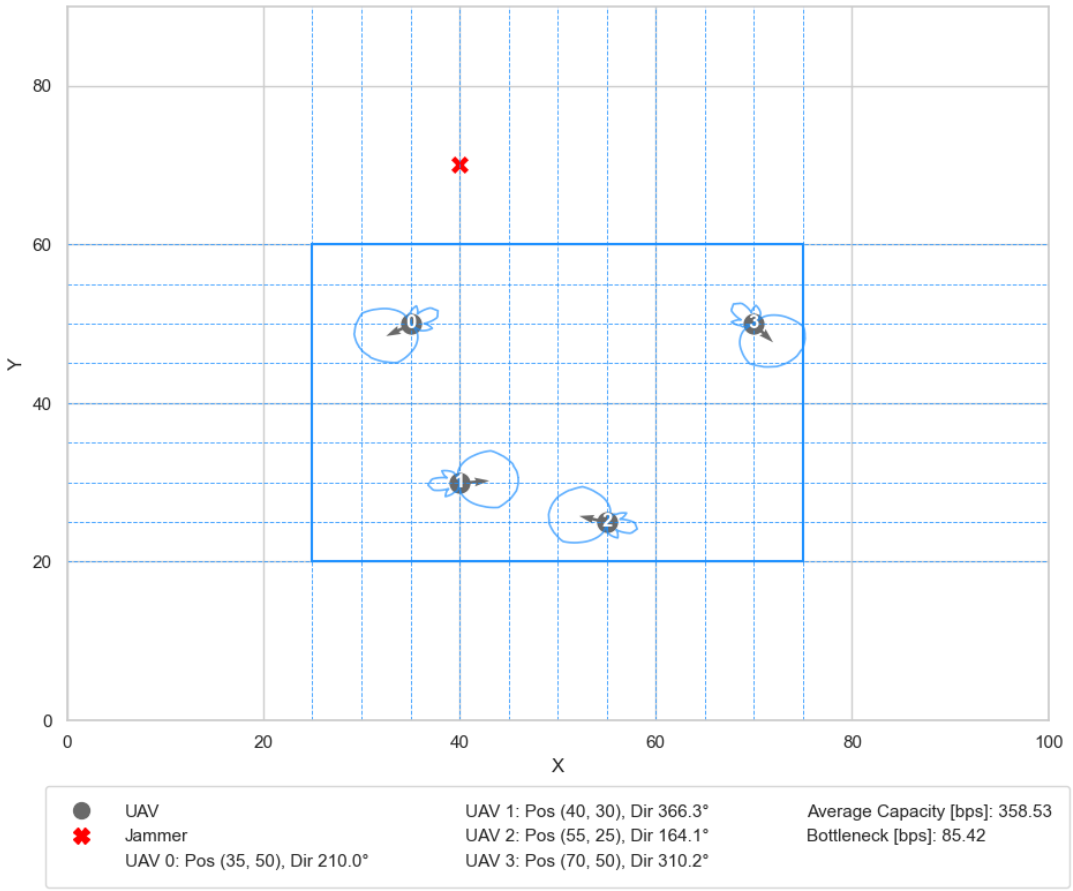


Figure 19: Directions established by implementing the PPO algorithm

Because of these results, a previously categorised training data set with 30000 different scenarios was included. The data set entered was structured as shown in Table 9.

Table 9: Format of the algorithm's training data set.

Jammer Position	**Directions Established**	**Reward**
[76.3, 15.7]	[210.6, 257.3, 167.1]	18851.4
[89.3, 48.0]	[43.4, 211.8, 186.7]	347672.2
[7.1, 62.5]	[115.8, 173.7, 305.6]	321831.0
[5.2, 56.6]	[239.7, 164.9, 279.7]	408192.0
...	...	...

For the following parameterisation:

Table 10: Parameters used in the PPO algorithm.

Parameter	Value
Timesteps	30000
Learning rate	$3e^{-2}$
Number of steps	2048
Batch size	64
Gamma	0.99
Clip range	0.2

Once the agent had been trained with the data above set, the model was saved and could be used to make predictions of optimal solutions in a much shorter execution time than that seen with GA.

The evolution curve of the average reward over the course of the training showed an evolutionary behaviour, stabilising at approximately 23000 *timesteps*, as can be seen in Figure 20.

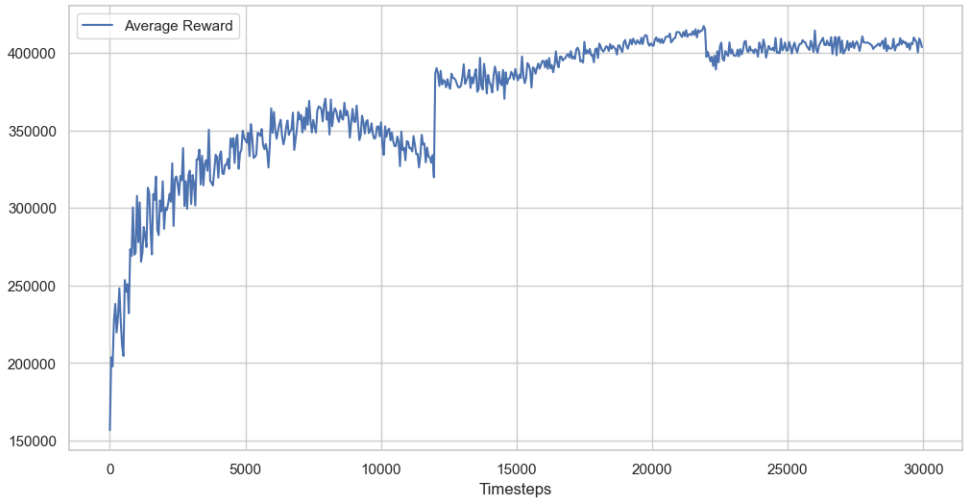


Figure 20: Evolution of the average reward value during training of the PPO algorithm with data set.

By running it, we obtained the following result, significantly better than the result obtained previously with GA, but with an execution time of less than 2 seconds.

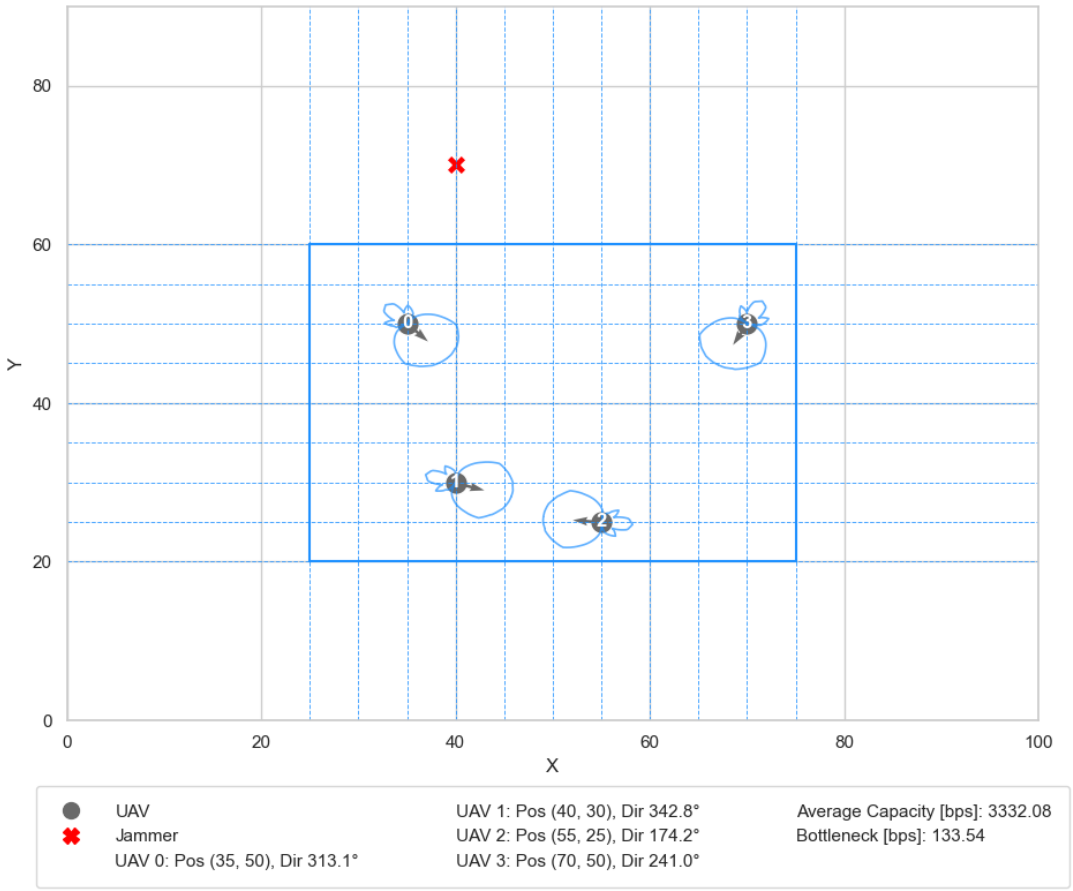
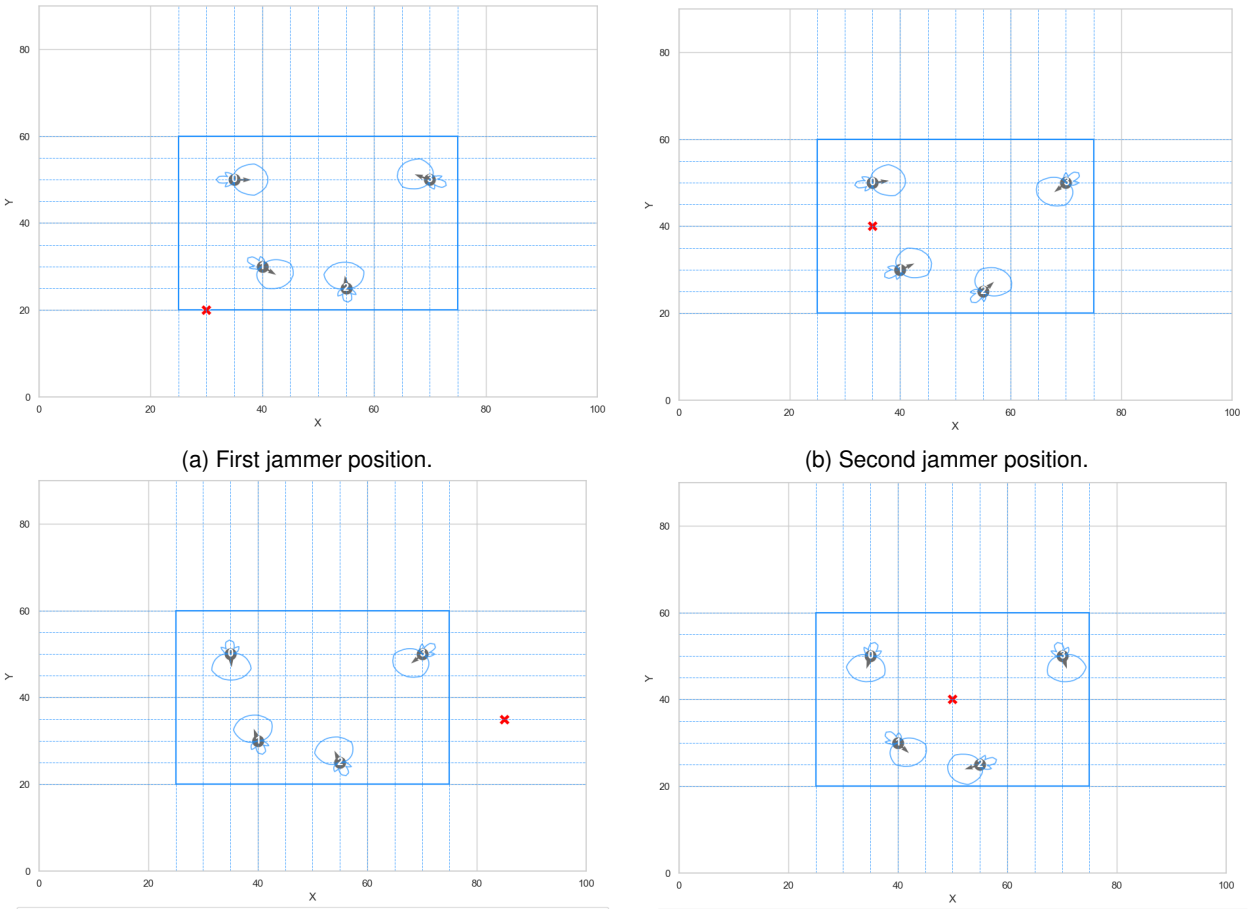


Figure 21: Result obtained by implementing the data set in the PPO algorithm.

The validation of this result, when compared with the previous result, allowed other topologies to be evaluated where the position of the jammer was variable. The results obtained for various jammer positions can be seen in Figure 22.



(a) First jammer position.

(b) Second jammer position.

(c) Third jammer position.

(d) Fourth jammer position.

Figure 22: Results of the algorithm for different jammer positions.

The results show that each UAV directs its antenna so that the jammer only has contact with the area with the lowest antenna radiation gain, assigning the lobes with the highest gain in the direction favouring greater reception capacity for the other UAVs.

4.5.3 Result obtained using the SAC algorithm

After analyzing the results obtained using the PPO algorithm, the results of the SAC algorithm were analyzed and compared.

The study of the training curve as a reflection of the effectiveness of the parameterisation began. Initially, a result was obtained that demonstrated the algorithm's inability to explore promising actions and correctly update the policy, as can be analysed in Figure 23.

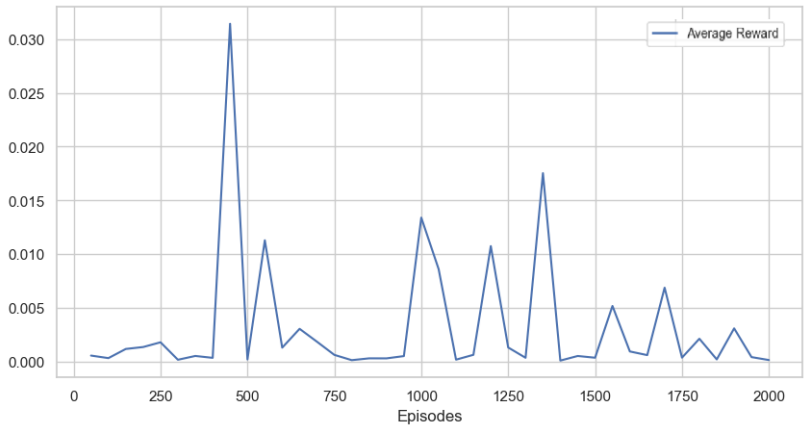


Figure 23: Evolution of the average reward of the SAC algorithm.

The image shows that the algorithm occasionally finds a strategy that maximises the reward in specific episodes. However, this strategy is not robust, and the reward decreases again.

Since this algorithm works directly with the *Q-values* learnt by the *Critic* layers to update the policy, the *Q-values* and, consequently, the rewards have a direct impact on the magnitude of the gradients used to optimise the policy.

The reward value was normalised to mitigate the frequency of *outliers* and achieve more stable training. This need arises from the wide range of values that the reward can assume. This range is easily seen in the previous results.

The normalisation carried out, Min-Max, can be described by the Equation 22.

$$x_{\text{normalised}} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{22}$$

Where x corresponds to the original reward value, x_{min} represents the minimum value of the verified rewards and finally, x_{max} symbolises the maximum value of the respective rewards.

For the reason above, normalisation proved to be effective in guiding training. Despite the low values, the increasing reward curve shows that training tends to impose a policy that maximises the reward compared to previous ones.

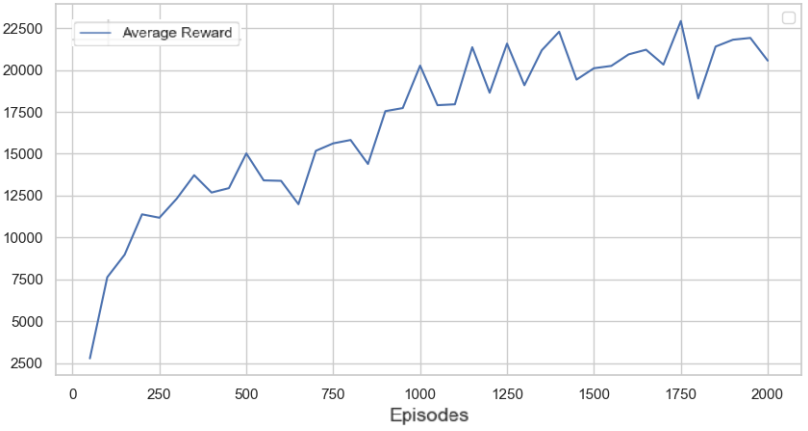


Figure 24: Training results after implementing reward normalisation.

With the proper adjustment of the hyperparameters from Table 11

Table 11: Hyperparameters used in the SAC algorithm

Hyperparameter	**Value**	**Description**
Actor Learning Rate	1e-1	Learning rate for the actor's network.
Critic Learning Rate	1e-1	Learning rate for the *Critic 1* and *Critic 2* networks.
Gamma (γ)	0.99	Discount factor for future rewards.
Tau (τ)	0.001	Refresh rate of target networks *Critic 1* and *Critic 2*.
Alpha (α)	0.01	Entropy coefficient that controls the degree of exploitation.
Replay Buffer Size	10,000	Maximum capacity of the *buffer* to store past experiences.
Batch Size	1024	Number of samples used for each gradient update.

After adjusting these parameters, the most significant result was obtained, which can be confirmed in Figure 25.

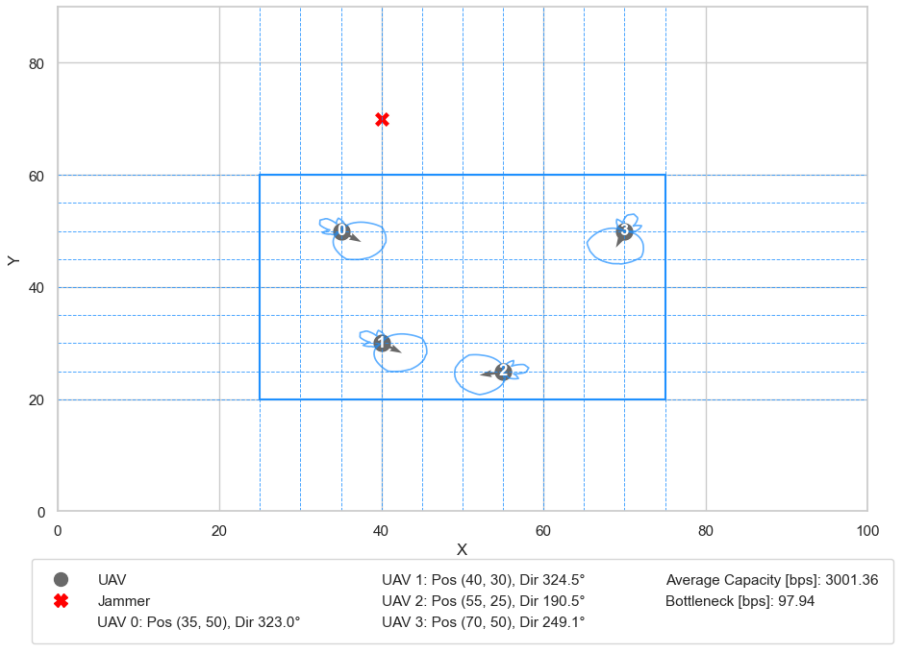
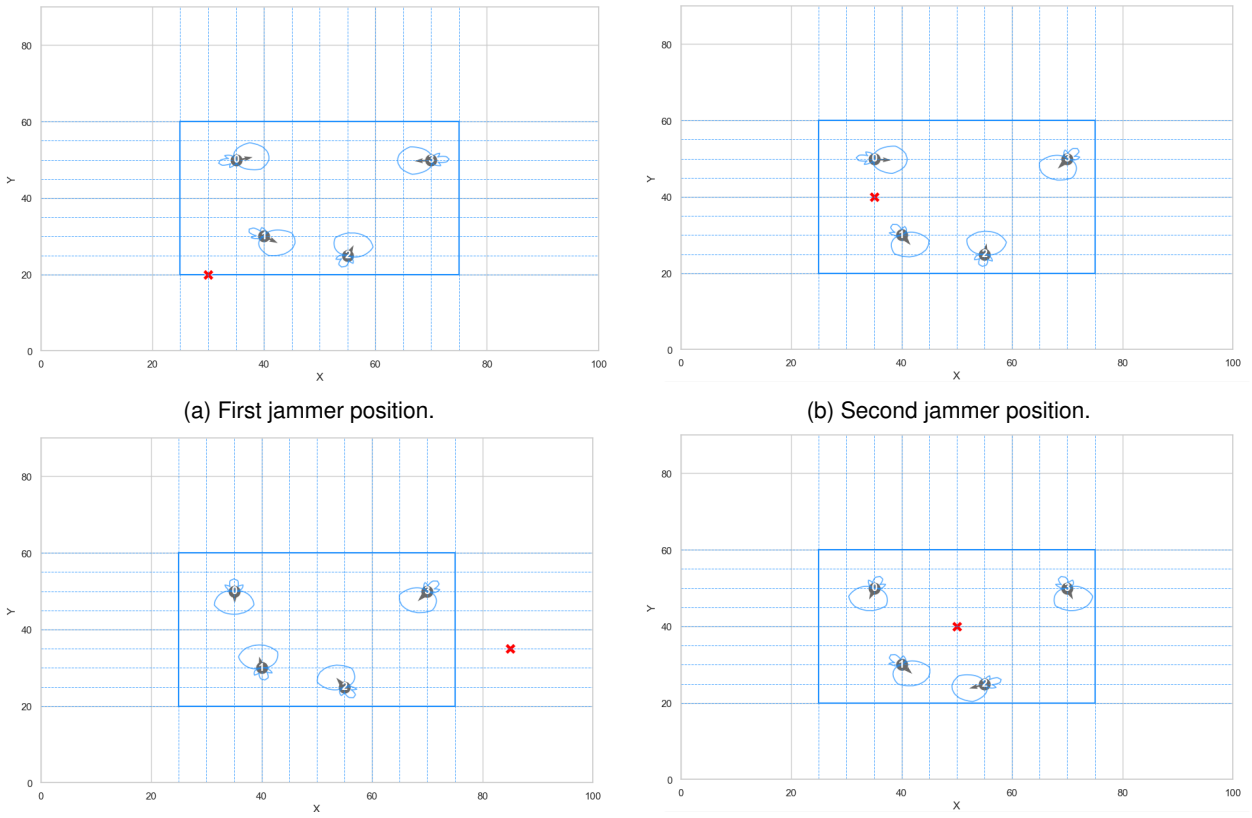


Figure 25: Evolution of the average reward of the SAC algorithm after implementing all the parameters.

The results obtained for various jammer positions can be seen in Figure 26.



(a) First jammer position.

(b) Second jammer position.

(c) Third jammer position.

(d) Fourth jammer position.

Figure 26: Results of the algorithm for different jammer positions.

Looking at the results in the figure, the configurations generated by the algorithm are very similar to those obtained by the PPO algorithm. This observation reflects that both algorithms were able to extract convergent policies.

Once these results were achieved, a new degree of freedom was added to the swarm: the position of each UAV.

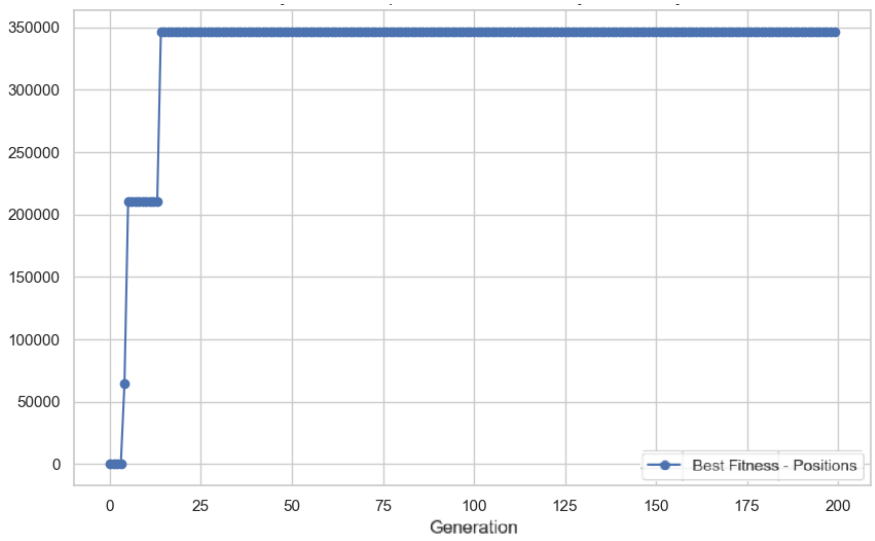
4.6 Scenario 2: Static Scenario with Exploitation of Antenna Directionality and UAV Positions

By applying this scenario, it was possible to study the effectiveness of the capabilities of RL algorithms in acquiring not only the directions but also the positions adjusted individually by each UAV to acquire a topology that maximises the transmission capabilities between the swarm.

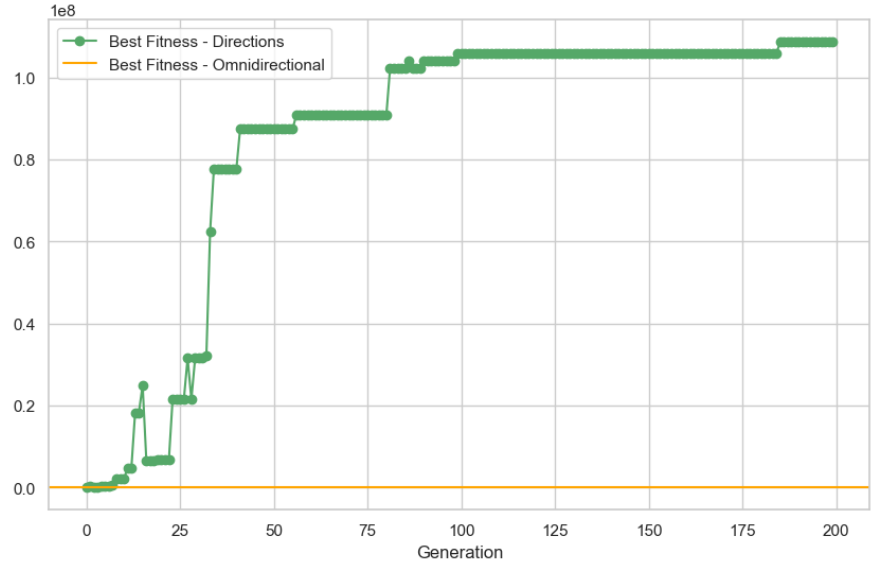
4.6.1 Result obtained using GA

Initially, the GA seeks to find positions favouring communications between UAVs. These results are then directed to the second part, where the GA aims to find the best directions for the directional antennas for the positions previously found.

Figure 27 shows that, given the lower complexity of the possible positions, the OF value converged more stably and much more quickly compared to the second part of the GA, where the directions are determined.



(a) Convergence of the OF in the acquisition of positions.



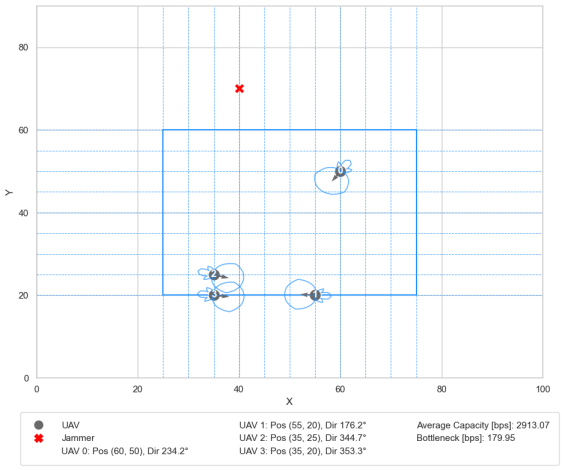
(b) Convergence of the OF in the acquisition of directions.

Figure 27: Convergence of OF of the encapsulated GA.

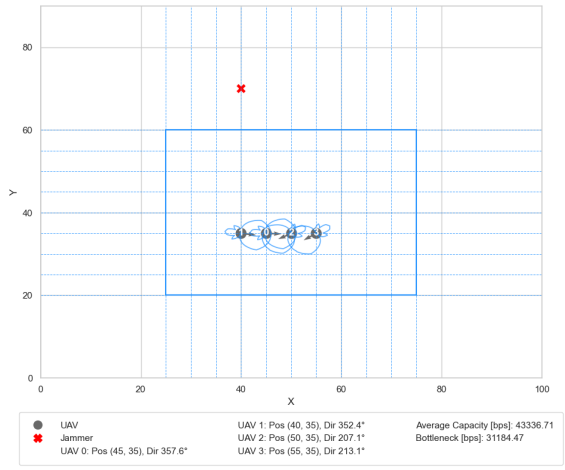
After analyzing the image, one can see the big difference between the OF value when applying directional antennas ($1.08e^8$) and omnidirectional antennas (1587).

After adjusting the parameterisation, with a significant increase in the population number, the performance was analysed according to the simulation time. Consequently, two limits were set for the simulation time: 15 and 60 seconds.

The results obtained are shown in Figure 28.



(a) Result obtained with 13 seconds of simulation.



(b) Result obtained with 52 seconds of simulation.

Figure 28: GA performance according to simulation time.

In the omnidirectional communication configuration, GA can obtain a very fine-tuned arrangement of the UAVs in a significantly shorter time interval. Its results are shown in Figure 29.

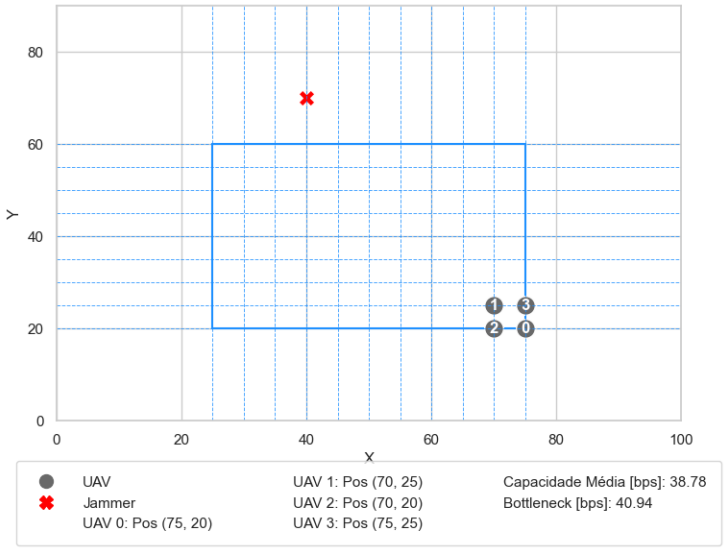


Figure 29: Result obtained with 23 seconds of simulation with omnidirectional antennas.

Analyzing the results highlighted above, it is clear that the GA seeks to acquire positions further away from the jammer to reduce the interference felt, and, on the other hand, it aims to bring the UAVs closer together to increase the power received by each node.

These results could be better since, to increase the objective function's value, the GA needs a time interval that is often unfeasible in a realistic operating environment.

Although the positioning of the UAVs using the directional antenna configuration is not ideal when compared to the omnidirectional configuration, there is a drastic decrease in communication performance when using the omnidirectional configuration.

4.6.2 Result obtained using PPO algorithm

Compared to the previous scenario, the algorithm was also trained using an external data set with 60000 randomly generated scenarios. However, this data set required new parameters: the Cartesian position of each UAV.

The data set was then structured as shown in Table 12.

Table 12: PPO algorithm training data set format.

Jammer Position	Directions	UAV Positions	Reward
[76.3, 15.7]	[210.6, 257.3, 167.1]	[35, 20, 45, 30, 55, 40]	18851.4
[89.3, 48.0]	[43.4, 211.8, 186.7]	[30, 50, 20, 60, 70, 30]	347672.2
[7.1, 62.5]	[115.8, 173.7, 305.6]	[40, 25, 30, 55, 45, 20]	321831.0
[5.2, 56.6]	[239.7, 164.9, 279.7]	[25, 35, 45, 55, 60, 70]	408192.0
...	...	...	...

The parameters that gave the best results after various tests were adjusted.

Table 13: Parameters used in the PPO algorithm.

Parameter	Value
Timesteps	60000
Learning rate	$3e^{-1}$
Number of steps	2048
Batch size	256
Gamma	0.99
Clip range	0.2

The evolution of the algorithm's average reward during the training process is shown in Figure 30.

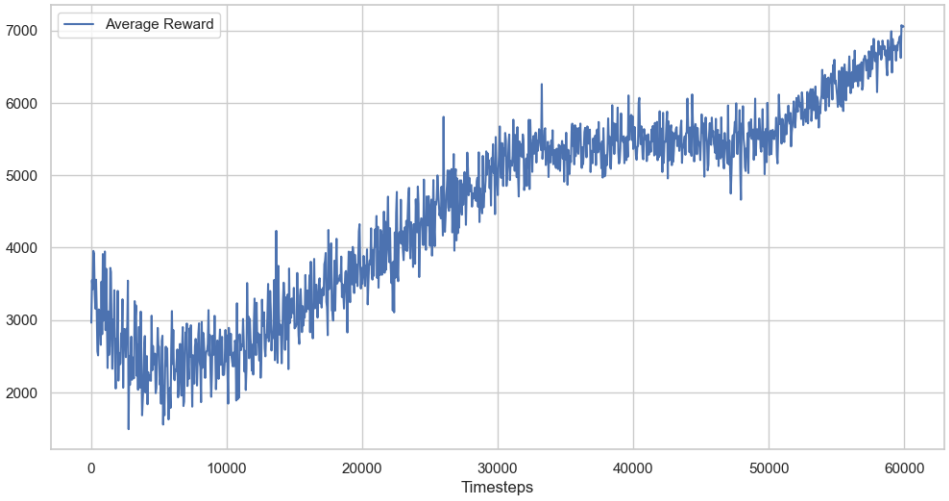
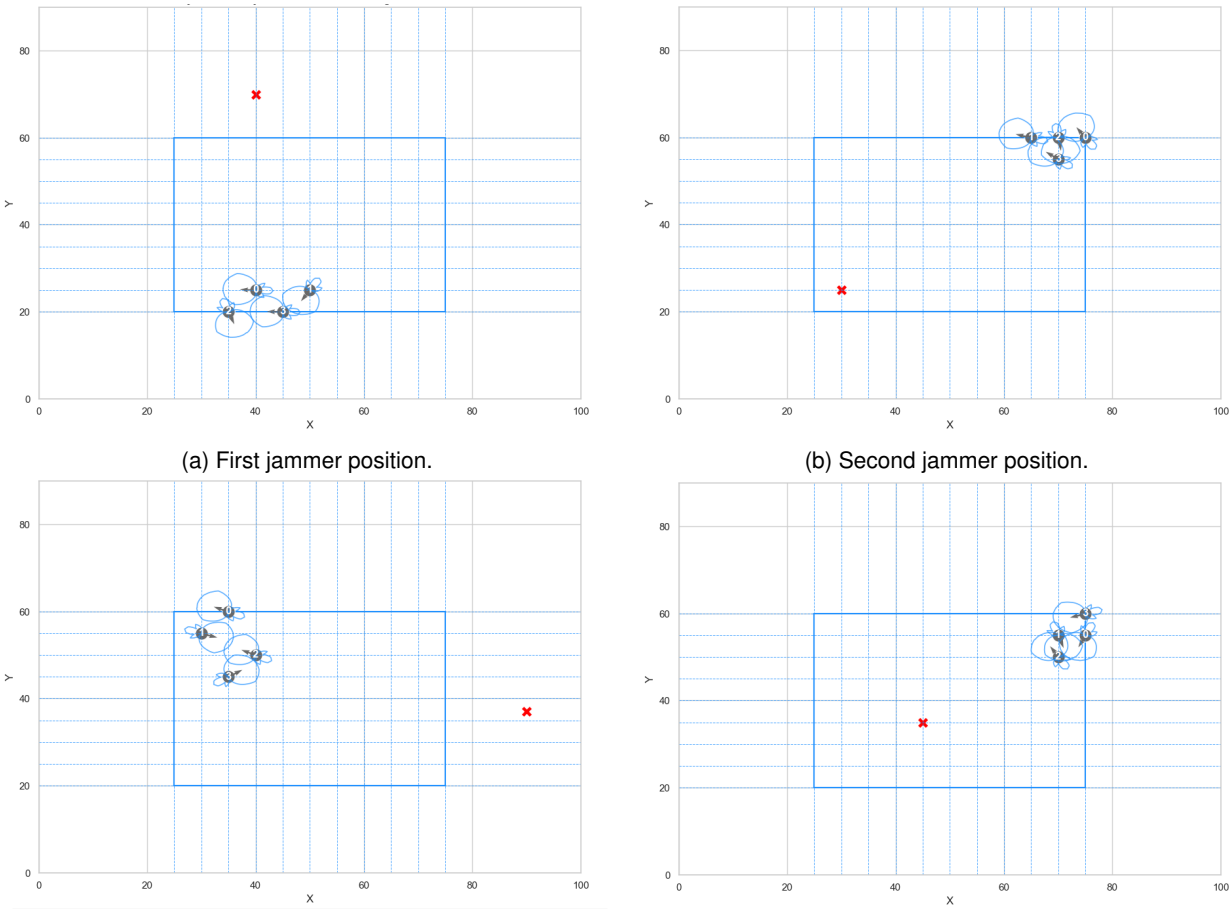


Figure 30: Evolution of the average reward value during training of the PPO algorithm with data set.

Analysing the evolution of the reward curve shows much instability in the agent's training process, reaching

relatively low values.

However, the algorithm's performance in response to jammer interference was analysed differently. The best results are shown in Figure 31.



(a) First jammer position.

(b) Second jammer position.

(c) Third jammer position.

(d) Fourth jammer position.

Figure 31: Algorithm results for different jammer positions.

A general analysis shows that the implemented algorithm was able to extract the positions with some effectiveness (it places the UAVs at the furthest end of the jammer and tends to reduce the distances between them), but the acquisition of the antenna directions proved to be entirely unsuitable for increasing the effectiveness of the swarm's communications.

Based on the analysis, it can be concluded that the algorithm is converging towards sub-optimal solutions. In contrast with PPO, SAC encourages the exploitation of the policy's entropy, unlike PPO, which tries to optimise the policy in small steps.

4.6.3 Result obtained using SAC algorithm

Following the same approach as in the previous scenario with the SAC algorithm, it was necessary to adjust its hyperparameters.

The adjustments counteracted the instability caused by the significant increase in the complexity of the scenario. Thus, when compared to the hyperparameters defined in the first scenario (Table 11), the following changes

were made:

- Reduction of the Actor and Critic learning rate to adjust the update weights.
- Increase the alpha value to promote greater exploration of the environment.
- Increase the replay buffer to increase the number of reused experiences, increasing training efficiency.
- Increase the batch size to promote greater stability.

The new hyperparameters can be found in Table 14

Table 14: Hyperparameters used in the SAC algorithm

Hyperparameter	**Value**	**Description**
Actor Learning Rate	1e-3	Learning rate for the actor's network.
Critic Learning Rate	1e-3	Learning rate for the *Critic 1* and *Critic 2* networks.
Gamma (γ)	0.99	Discount factor for future rewards.
Tau (τ)	0.01	Refresh rate of target networks *Critic 1* and *Critic 2*.
Alpha (α)	0.1	Entropy coefficient that controls the degree of exploitation.
Replay Buffer Size	100,000	Maximum capacity of the *buffer* to store past experiences.
Batch Size	2048	Number of samples used for each gradient update.

After adjusting these parameters, the most significant result was obtained, which can be confirmed in Figure 32.

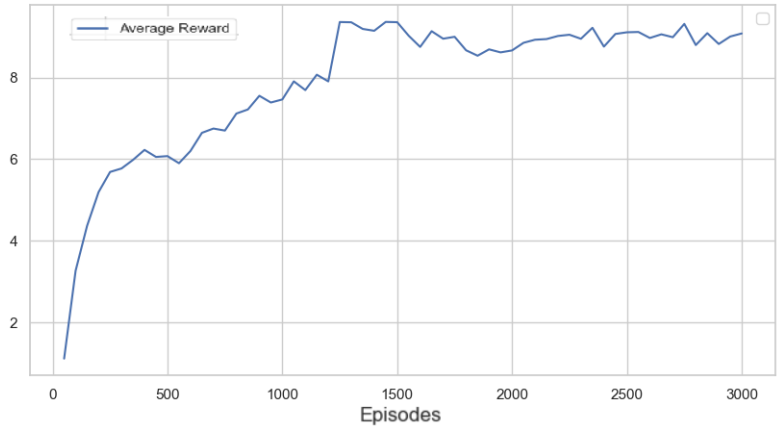
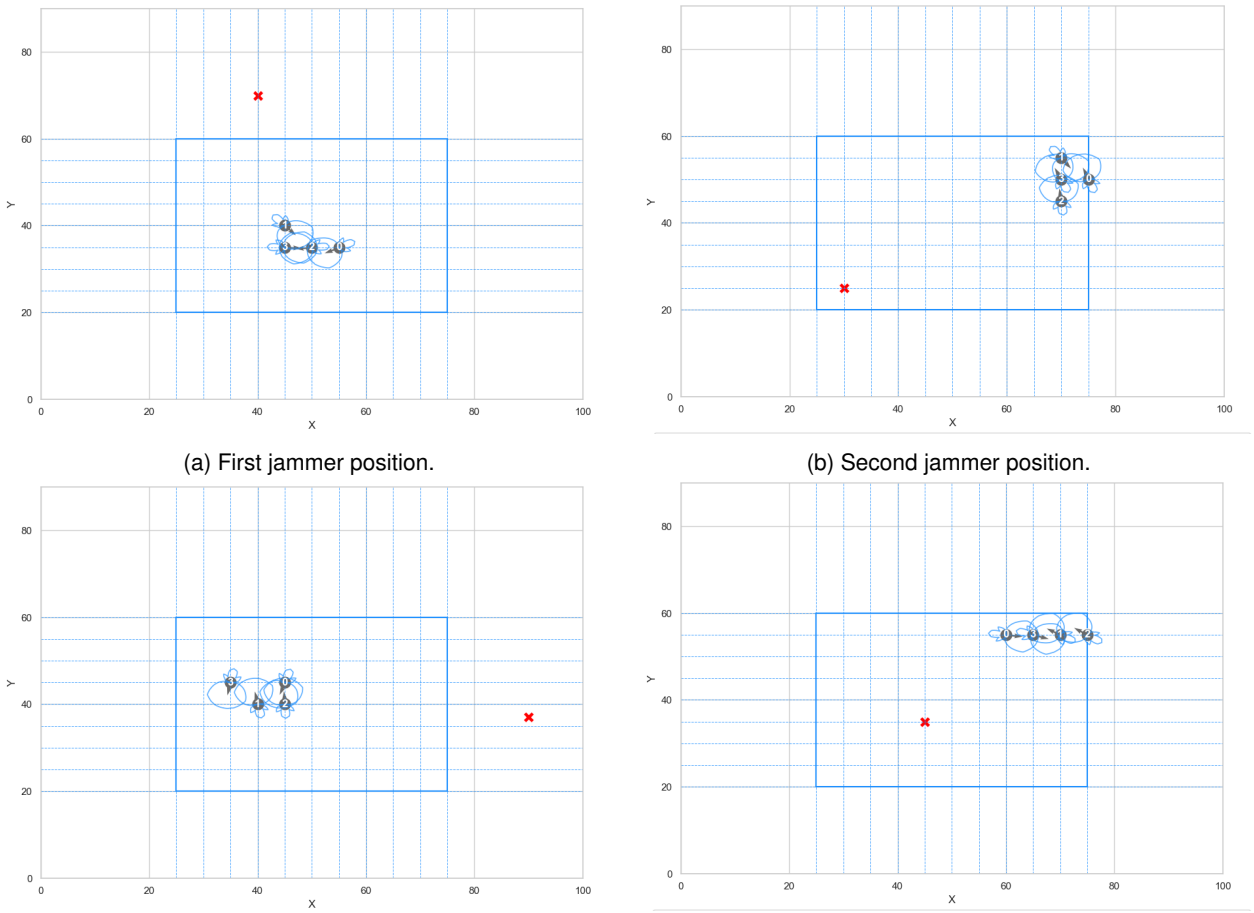


Figure 32: Evolution of the average reward of the SAC algorithm after implementing all the parameters.

Analysis of the image shows that the parameters defined initially helped promote exploration of the environment, where there is a rapid increase in the average reward values with stabilisation in the last episodes, where it is promoted that the policy undergoes minor adjustments to converge on high reward results.

Applying the model obtained after the 3000 training episodes, the results in Figure 33 were achieved, with a simulation time of around five seconds.



(a) First jammer position.

(b) Second jammer position.

(c) Third jammer position.

(d) Fourth jammer position.

Figure 33: Results of the SAC algorithm for different jammer positions.

A brief analysis shows that the SAC algorithm has achieved better results than the PPO since it reasonably presents a topology where the UAVs tend to move away from the jammer (also achieved previously). However, the assignment of antenna directions is achieved more efficiently in order to maximise communication capacity between the swarm's nodes.

4.7 Scenario 3: Swarm in Progression

In this scenario, it was necessary to adapt the previously evaluated algorithms so that it would be possible to sequentialise the position updates. As described in Section 3.1, it was necessary to carry out temporal discretisation to reduce the complexity of implementing the algorithms and analysing the results.

Figure 34 shows the flowchart of the system's execution.

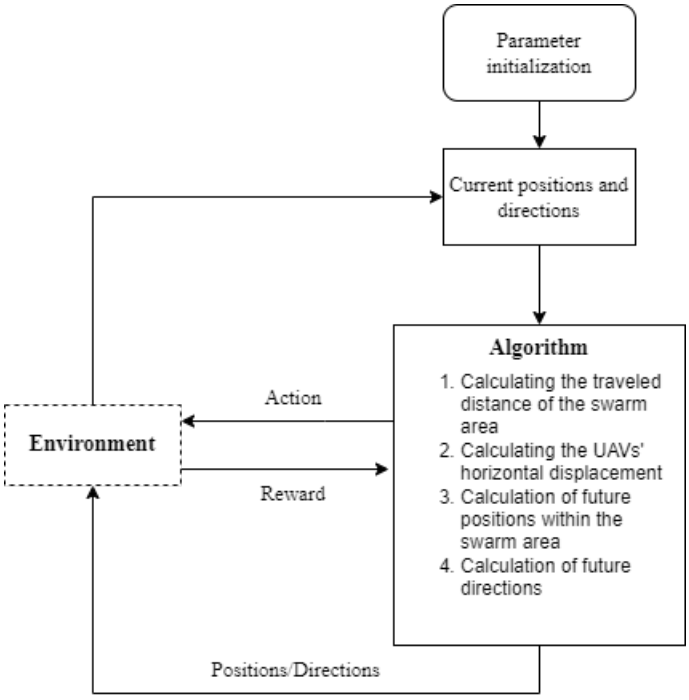


Figure 34: System execution flowchart.

Thus, the following were added to the parameters, defined in Table 15.

Table 15: Parameters added to the UAV swarm

Parameter	Value
Refresh interval	7.5 s
UAV speed	2 m/s
Direction of progression	Axis of xx

Given the poor results of the PPO algorithm in the second scenario and its consequent outperformance by the SAC algorithm, the RL approach was based solely on analyzing the results of the SAC algorithm in this scenario.

4.7.1 Result obtained using GA

As previously discussed, it was necessary to explore the GA parameterisation carefully. Since the topology update was set to a value of $7.5\ s$, the time interval needed by the algorithm to determine the new positions and directions had to be shorter than this update interval.

Consequently, experimentation found that for a population of 14 chromosomes over 40 generations, results were achieved with an average simulation time of $6.43s$.

From the initial topology in Figure 35.

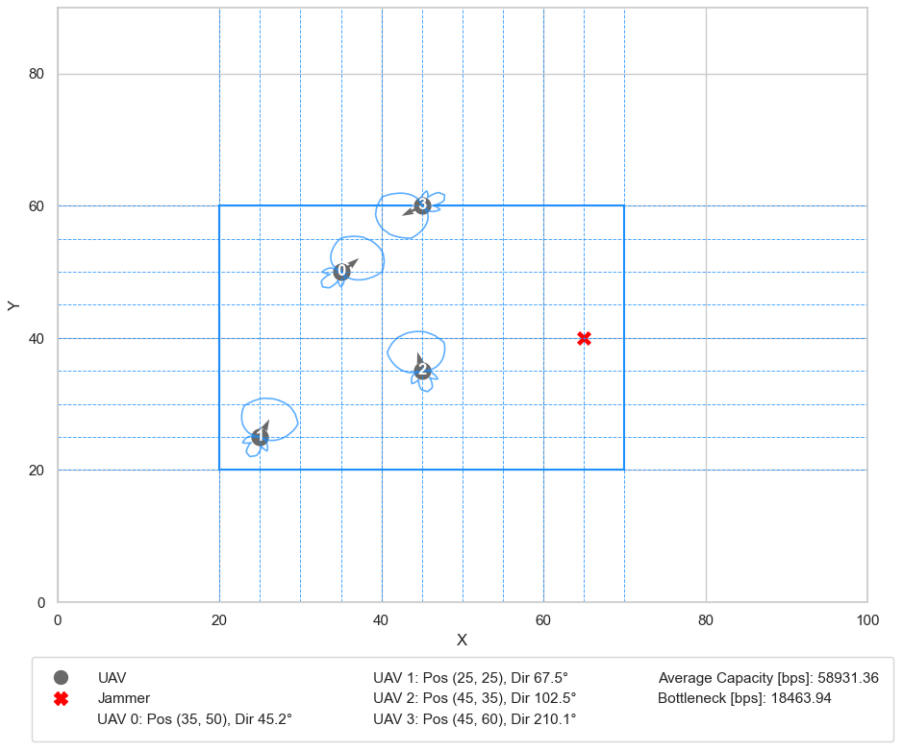
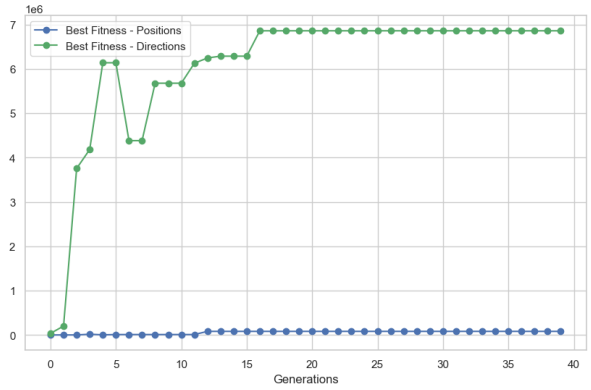
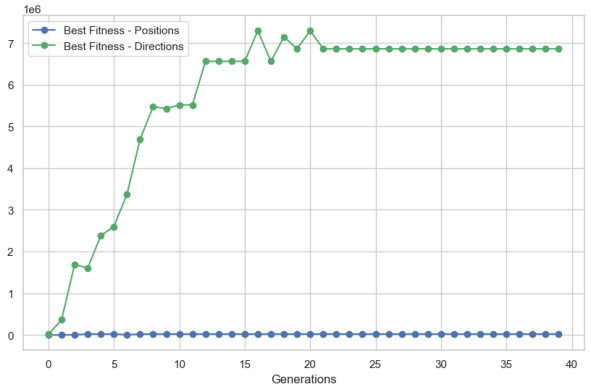


Figure 35: Topology at simulation time $0.0s$.

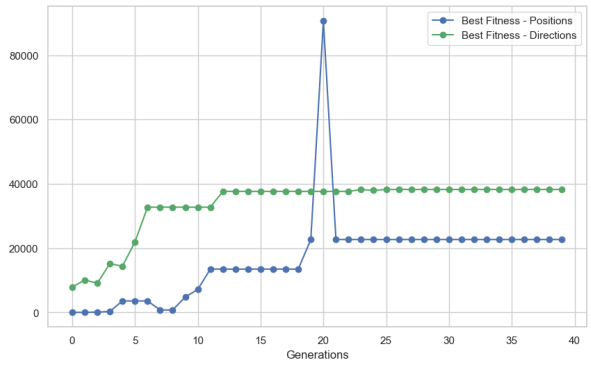
The simulation obtained the evolution curves of the OF values, shown in Figure 36.



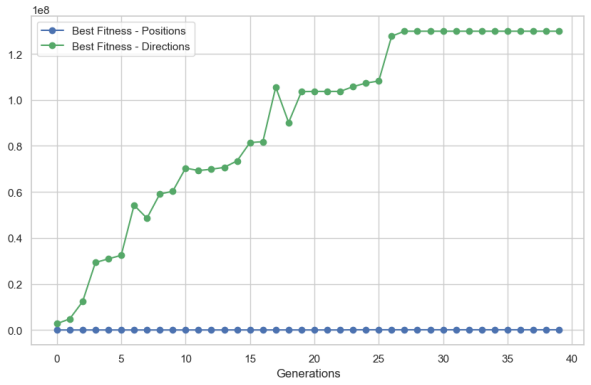
(a) Evolution of the OF values at 7.5 seconds of simulation time.



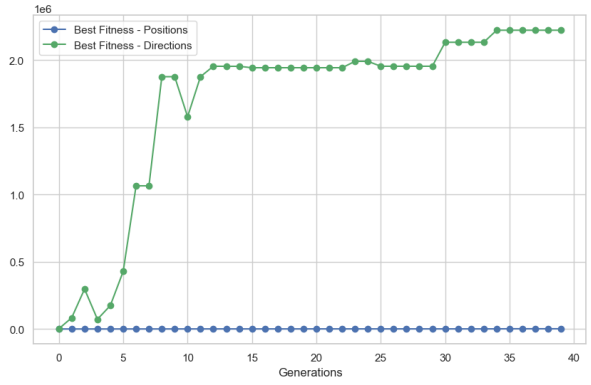
(b) Evolution of the OF values at 15.0 seconds of simulation time.



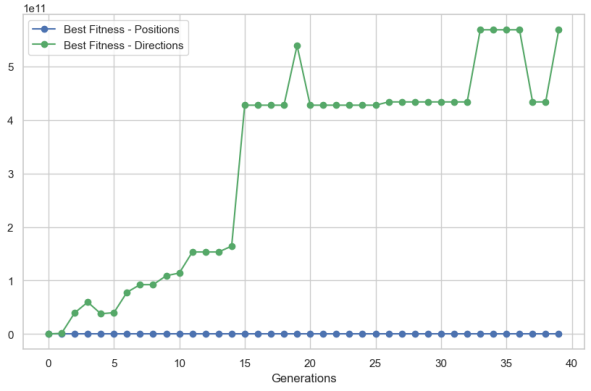
(c) Evolution of the OF values at 22.5 seconds of simulation time.



(d) Evolution of the OF values at 30.0 seconds of simulation time.



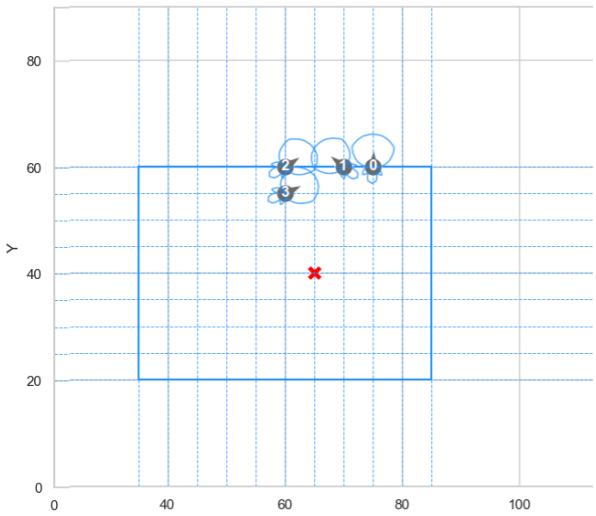
(e) Evolution of the OF values at 37.5 seconds of simulation time.



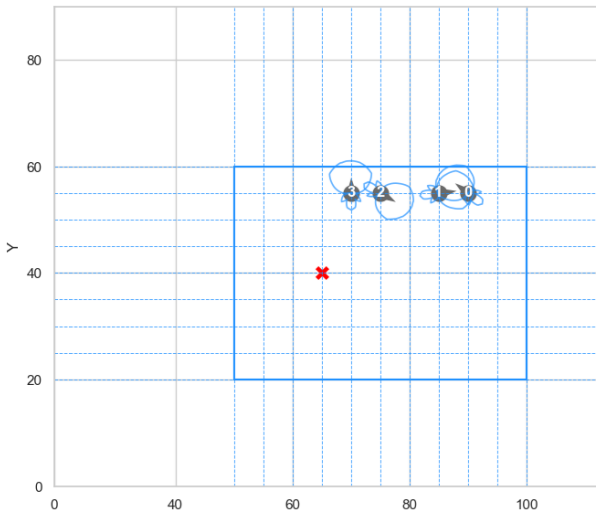
(f) Evolution of the OF values at 45.0 seconds of simulation time.

Figure 36: Evolution of the OF values for different simulation times.

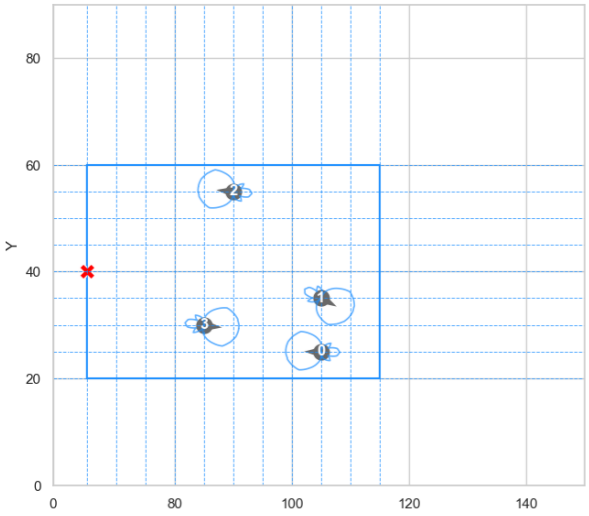
The results of each simulation can be seen in Figure 37.



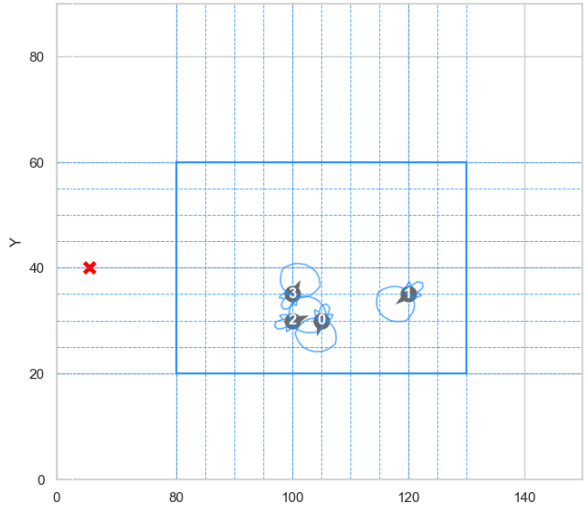
(a) Topology at 7.5 seconds of simulation time.



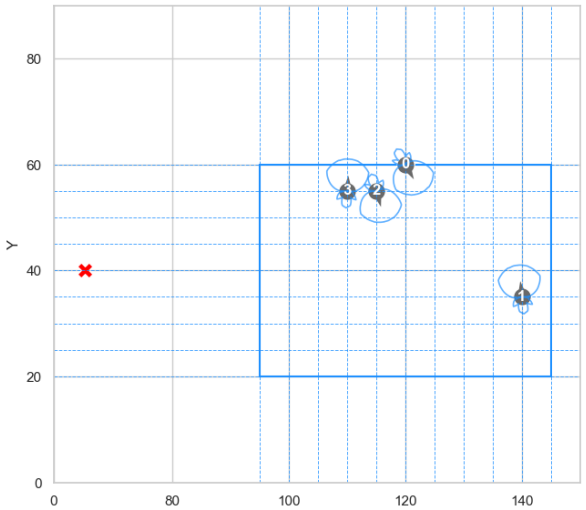
(b) Topology at 15.0 seconds of simulation time.



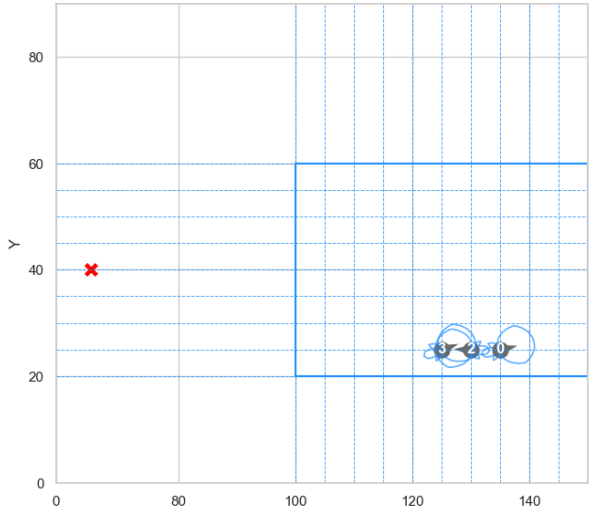
(c) Topology at 22.5 seconds of simulation time.



(d) Topology at 30.0 seconds of simulation time.



(e) Topology at 37.5 seconds of simulation time.



(f) Topology at 45.0 seconds of simulation time.

Figure 37: Results of the GA algorithm for different simulation times.

A brief analysis of the results shows that even with poor parameterisation, the GA tends to cluster the UAVs and direct the antennas as effectively as possible. Better results would be obtained if the maximum simulation time were extended, allowing the GA to explore positions and directions more thoroughly.

4.7.2 Result obtained using SAC algorithm

Since in-depth training and parameterisation exploration had already been carried out in the previous scenario, the SAC results for this scenario were analysed, starting from the initial topology specified in Figure 35.

The average reward curve shown in Figure 38 was obtained during the training.

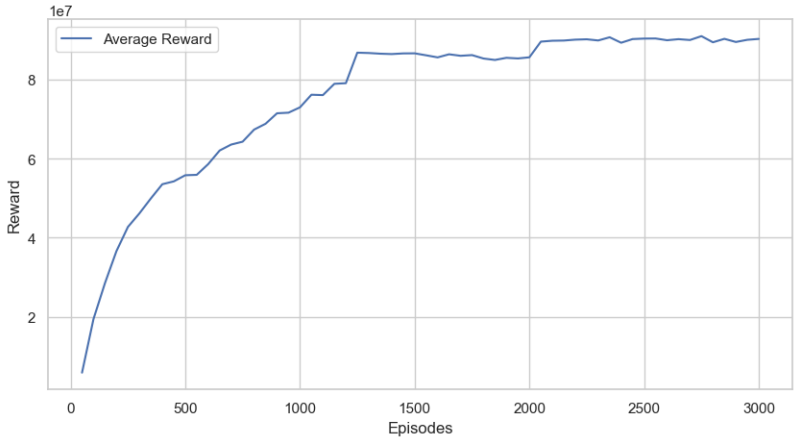
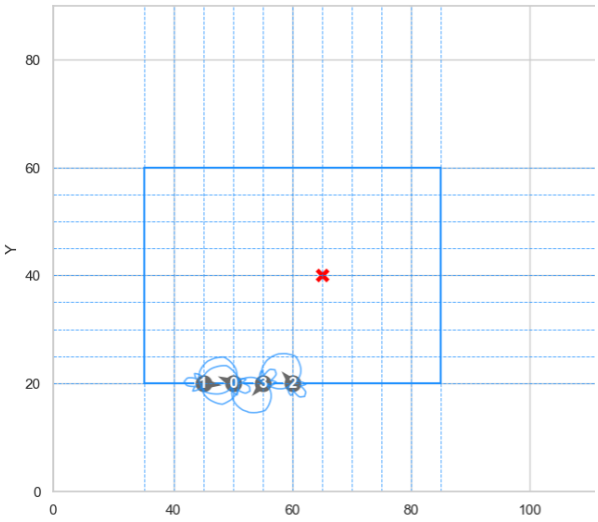


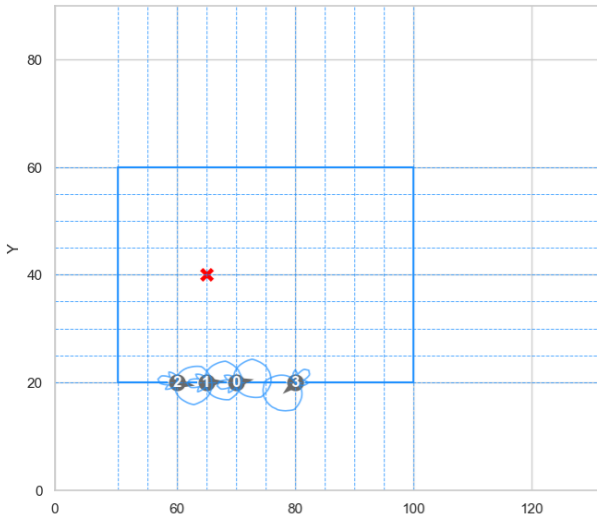
Figure 38: Evolution of the average reward of the SAC algorithm after implementing all the parameters.

For the training created, the results obtained by the SAC algorithm are shown in Figure 39.

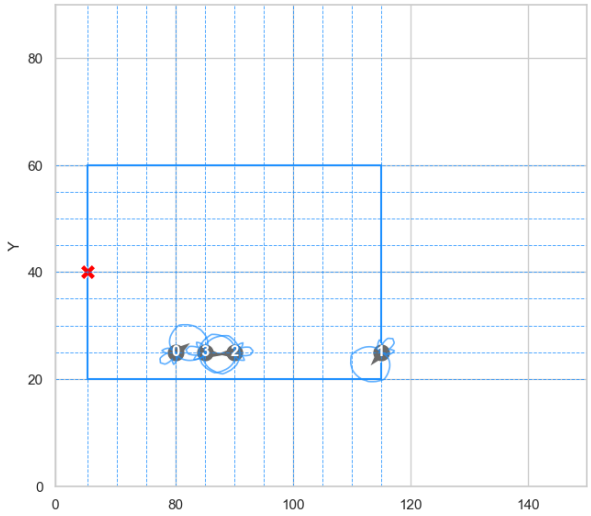
Analysing the results obtained by the SAC algorithm, it is notable that it performs better both in acquiring positions, where it exploits the depth of the sub-grade more effectively, and in acquiring directions. The results show a denser cluster of UAVs in the swarm, and the established directions are less dispersed.



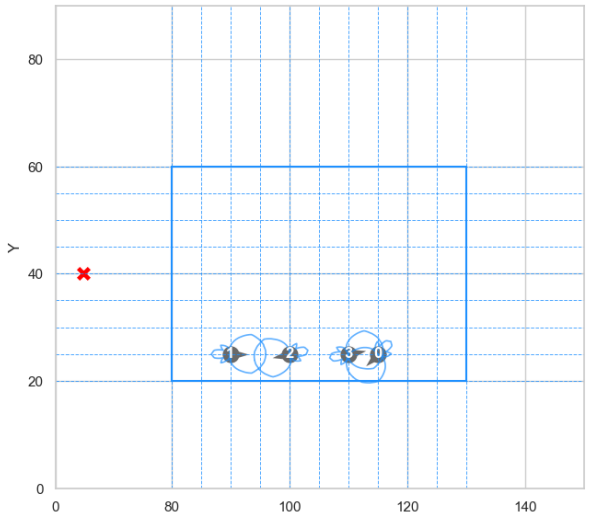
(a) Topology at 7.5 seconds of simulation time.



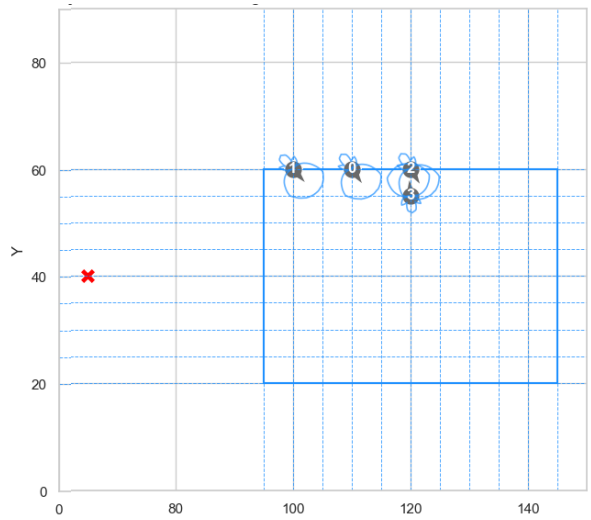
(b) Topology at 15.0 seconds of simulation time.



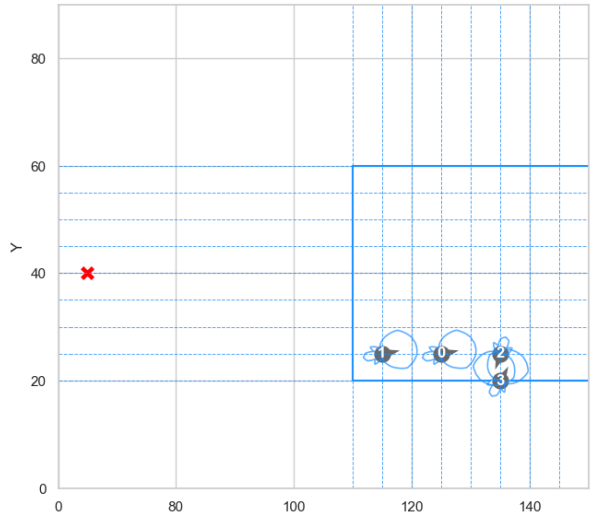
(c) Topology at 22.5 seconds of simulation time.



(d) Topology at 30.0 seconds of simulation time.



(e) Topology at 37.5 seconds of simulation time.



(f) Topology at 45.0 seconds of simulation time.

Figure 39: Results of the SAC algorithm for different simulation times.

Chapter 5

Conclusions and Future Work

This section will present the conclusions from the above results and discuss their relevance.

Possible proposals for future research will also be addressed to complement and continue the research carried out in this dissertation.

5.1 Conclusions

The main objective of this thesis is to investigate and evaluate the reliability of applying RL-based techniques to decision-making regarding anti-jamming measures within a swarm of UAVs. This study seeks to highlight the advantages and disadvantages found throughout the execution of the tests, which were divided into three distinct stages with a progressive increase in complexity.

The specific challenge of this thesis was to mitigate the effect of interference on communications and ensure that the swarm of UAVs continues to communicate and coordinate, even when a jamming beacon is present.

The solution presented in this thesis is using AI to provide the UAV swarm with collaborative anti-jamming behaviour, taking advantage of collective intelligence to predict and avoid imposed interference and successfully achieve mission objectives. To this end, directional antenna models and freedom of UAV positions are combined to improve communication capacity and mitigate the effect of interference. Two main types of algorithms have been developed:

GA is used as baseline to optimize the positions and directions of the UAVs' antennas to maximize communication capacity. With PPO and SAC, RL allows UAVs to learn to adjust the direction of the antennas and their movements dynamically to minimize the interference caused by the jammer.

For this, three experimental scenarios were developed, and two different approaches were compared: one based on a genetic algorithm and the other using RL techniques. In each scenario, the results of both approaches were analysed, allowing for a detailed assessment of their respective performances.

In the first scenario, where the positions of the UAVs remain static, and only the directions of the directional antennas are adjusted, the results indicate that the genetic approach offers an adjusted solution comparable to those obtained with the RL algorithm PPO and SAC . In addition, verifying the importance of adequately training the RL algorithm was possible, which required importing randomly generated topologies for the scenario being analysed and carefully adjusting the parameters in both approaches.

In the second scenario, the degree of complexity was significantly increased with the introduction of an additional degree of freedom for the individual movement of each UAV. Given this increase in complexity, the results show that the effectiveness of the genetic approach is directly related to the simulation time available for acquiring the optimal topology. On the other hand, the RL approach, using the Soft Actor-Critic algorithm, proved to be more effective, as the policy adopted by the agent was updated through direct interaction between the agent and the environment and by calculating the rewards estimated through Q-values.

In the third and final scenario, a new degree of freedom was introduced: the progressive movement of the swarm over time. At this stage, simulation time became a critical metric since the movement of the UAVs was discretized into time intervals (*time slots*), making it impossible for the simulation time to exceed this limit. The results show that although the genetic approach can explore different interference mitigation strategies, it cannot optimize the topology effectively. In contrast, the RL approach significantly improved results in reducing perceived interference within the same time interval.

In general, the GA approach produced efficient results, depending directly on the simulation time available for the algorithm. On the other hand, although the RL approach requires a more complex and time-consuming pre-training process, it provides faster and more accurate results since the model can be pre-trained for future scenarios. Thus, the RL-based approach is a promising and effective tool for obtaining optimized topologies to combat malicious interference in dynamic scenarios. In contrast, the GA approach is unfeasible in real-time, where solid topologies must be determined in the shortest possible time.

5.2 Future Work

Several potential improvements can be implemented in this research to resolve the limitations of the developed model and increase its overall functionality.

This research has developed a model that analyses and adjusts topologies discretely, simplifying and abstracting from the complexity of implementing the routes to follow.

Therefore, as future research, it is proposed to superimpose on this model the calculation of effective routes to prevent the swarm from splitting and to add a new metric: the time it takes for the swarm to adopt the topology determined in this model.

One assumption on which this research was based was that, given the position of the jammer, all the coordinates and directions adopted by each UAV were broadcast over the network instantaneously and free from interference.

Therefore, developing a protocol that adjusted the communication channels for disseminating indications generated by an external entity (ground station, for example) would be a valuable addition to the model developed in this dissertation.

In this investigation, the assumption was made that the location of the jammer was known in advance.

Jammer detection and localization represent an essential advance in anti-jamming, allowing UAVs to operate in more dynamic and unpredictable environments where the jammer's position is not predetermined.

To this end, several mechanisms could be developed, such as: equipping UAVs with radio frequency sensors to detect anomalies in the frequency spectrum, using triangulation techniques based on the interfered signal levels received by multiple UAVs to estimate the jammer's position, or developing collaborative algorithms between UAVs to share information from jammed signals and detect the jammer's location cooperatively.

UAVs equipped with null-steering antennas can detect the direction of the jammer or jammers and adjust the antenna's radiation pattern to cancel or minimize the jamming signal coming from those direction. This would increase the resilience of the communication system in scenarios where jammers are used to disrupt operations.

Null-steering antennas can be integrated into RL algorithms, where UAVs collectively predict areas of interference and adjust their trajectories and communications accordingly.

Future work could focus on developing algorithms that integrate null-steering antennas with machine learning or swarm intelligence techniques. This would allow UAVs to detect interference and respond dynamically in real-time, optimizing communication and avoiding jamming more effectively.

Annex A

Software Packages

This annex shows the table of software packages used, a brief description and their version.

Table 16: Software packages used, Part 1.

Package **Description**	**Version**
`gym` Library for reinforcement learning environments.	0.25.2
`gymnasium` Similar to `gym`, used for reinforcement learning environments.	0.29.1
`numpy` Library for mathematical calculations and array manipulation.	1.26.4
`seaborn` Data visualization library based on Matplotlib, focused on statistical plots.	0.13.2
`matplotlib.pyplot` Submodule of Matplotlib for data visualization and creating plots.	3.9.2
`pandas` Library for data manipulation and analysis in tabular form (DataFrames).	2.2.2
`matplotlib.lines.mlines` Module for drawing lines in Matplotlib plots.	3.9.2
`matplotlib.patches` Submodule for drawing geometric shapes.	3.9.2
`matplotlib.transforms` Used for performing geometric transformations.	3.9.2
`matplotlib.legend_handler` Customizes legends in Matplotlib plots.	3.9.2
`plotly.offline.iplot` Submodule for generating offline interactive plots.	5.9.0
`plotly.graph_objects` Module for creating customizable interactive graphs.	5.9.0
`plotly.express` Simplified interface for creating interactive plots.	5.9.0
`random` Module for generating random numbers.	1.2.4
`networkx` Library for creating and analyzing complex networks.	3.1

Table 17: Software packages used, Part 2.

Package **Description**	**Version**
`stable_baselines3.PPO` Proximal Policy Optimization (PPO) algorithm for reinforcement learning.	2.2.1
`stable_baselines3.A2C` Advantage Actor-Critic (A2C) algorithm for reinforcement learning.	2.2.1
`stable_baselines3.common.env_checker` Utility to check the compatibility of reinforcement learning environments.	2.2.1
`stable_baselines3.common.evaluation` Utility for evaluating trained policies in reinforcement learning.	2.2.1
`stable_baselines3.common.monitor` Used for monitoring the performance of algorithms in environments.	2.2.1
`stable_baselines3.common.vec_env.SubprocVecEnv` Creates parallelized simulation environments for more efficient learning.	2.2.1
`stable_baselines3.common.vec_env.DummyVecEnv` Creates non-parallelized simulation environments.	2.2.1
`stable_baselines3.common.callbacks.BaseCallback` Module for customizing callbacks during training in reinforcement learning.	2.2.1
`torch` (`torch as th`) Tensor computation and deep learning library.	2.2.1
`itertools` Standard library for handling iterators and combinatorics in Python.	10.1.0
`sklearn.preprocessing` Submodule for data preprocessing in Scikit-learn.	1.2.2
`time` Standard library for time-related functions.	1.5.5

Annex B

Simulation Time and Device Specifications

This annex sets out the average durations of the simulations carried out in this thesis, as well as the characteristics of the device used.

Device Specifications

- **Processor:** AMD Ryzen 7 6800HS with Radeon Graphics, 3.20 GHz
- **RAM Memory:** 16.0 GB (15.3 GB usable)
- **System type:** 64-bit operating system, x64-based processor

Average duration of simulations

During the execution of the simulations, the total duration and average duration of each simulation were measured. The table below summarizes the results obtained:

Table 18: Average Simulation Times for GA (Encapsulated)

Population	Generations	Average Time [min:sec]
50	100	02:30
100	100	03:40
200	100	09:10
100	200	08:40

Table 19: Average Times for PPO Training

Timestep	Average Time [min:sec]
1000	08:00
10000	90:00
50000	450:00
100000	900:00
200000	1800:00

Table 20: Average Times for SAC Training

Timestep	Average Time [min:sec]
1000	10:00
10000	100:00
50000	500:00
100000	1000:00
200000	20000:00

References

[1] Juhyun Kim et al. “Real-time UAV sound detection and analysis system”. In: *2017 IEEE Sensors Applications Symposium (SAS)*. IEEE. 2017, pp. 1–5.

[2] Kazi Tanvir Ahmed Siddiqui et al. “Development of a swarm uav simulator integrating realistic motion control models for disaster operations”. In: *arXiv preprint arXiv:1704.07335* (2017).

[3] Matthias R Brust and Bogdan M Strimbu. “A networked swarm model for UAV deployment in the assessment of forest environments”. In: *2015 IEEE Tenth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*. IEEE. 2015, pp. 1–6.

[4] Heorhii Dementiiuk et al. “Analysis of the destructive impact of attack drones on critical civil infrastructure: a combined method of protection based on the application of an electromagnetic shield”. In: *Scandinavian Journal of Information Systems* 35.1 (2023), pp. 29–37.

[5] Kerry Chávez. “Learning on the fly: Drones in the Russian-Ukrainian war”. In: *Arms Control Today* 53.1 (2023), pp. 6–11.

[6] Karel Pärlin, Muhammad Mahtab Alam, and Yannick Le Moullec. “Jamming of UAV remote control systems using software defined radio”. In: *2018 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE. 2018, pp. 1–6.

[7] Yanqi Zhang, Bo Zhang, and Xiaodong Yi. “Adaptive data sharing algorithm for aerial swarm coordination in heterogeneous network environments (short paper)”. In: *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer. 2018, pp. 202–210.

[8] Xiaohui Li and Andrey V Savkin. “Networked unmanned aerial vehicles for surveillance and monitoring: A survey”. In: *Future Internet* 13.7 (2021), p. 174.

[9] Gheorghe Udeanu, Alexandra Dobrescu, and Mihaela Oltean. “Unmanned aerial vehicle in military operations”. In: *Sci. Res. Educ. Air Force* 18.1 (2016), pp. 199–206.

[10] Kanika Grover, Alvin Lim, and Qing Yang. “Jamming and anti–jamming techniques in wireless networks: a survey”. In: *International Journal of Ad Hoc and Ubiquitous Computing* 17.4 (2014), pp. 197–215.

[11] Aristides Mpitziopoulos et al. “A survey on jamming attacks and countermeasures in WSNs”. In: *IEEE communications surveys & tutorials* 11.4 (2009), pp. 42–56.

[12] Hong-Ning Dai et al. “On the capacity of multi-channel wireless networks using directional antennas”. In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, pp. 628–636.

[13] António Fernando Alves Carneiro et al. “Smart Antenna for Application in UAVs”. In: *Information* 9.12 (2018), p. 328.

[14] Irfan Ahmed et al. “A survey on hybrid beamforming techniques in 5G: Architecture and system model perspectives”. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3060–3097.

[15] Shajahan Kutty and Debarati Sen. “Beamforming for millimeter wave communications: An inclusive survey”. In: *IEEE communications surveys & tutorials* 18.2 (2015), pp. 949–973.

[16] Ashish Srivastava and Jay Prakash. “Future FANET with application and enabling techniques: Anatomization and sustainability issues”. In: *Computer science review* 39 (2021), p. 100359.

[17] Ilker Bekmezci, Ozgur Koray Sahingoz, and Şamil Temel. “Flying ad-hoc networks (FANETs): A survey”. In: *Ad Hoc Networks* 11.3 (2013), pp. 1254–1270.

[18] Demeke Shumeye Lakew et al. “Routing in flying ad hoc networks: A comprehensive survey”. In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 1071–1120.

[19] Muhammad Asghar Khan et al. “Dynamic routing in flying ad-hoc networks using topology-based routing protocols”. In: *Drones* 2.3 (2018), p. 27.

[20] Altaf Hussain et al. “Co-DLSA: Cooperative delay and link stability aware with relay strategy routing protocol for flying Ad-hoc network”. In: *Human-centric Computing and Information Sciences* 12.34 (2022), pp. 428–447.

[21] Subhrapratim Nath et al. “Optimizing FANET routing using a hybrid approach of firefly algorithm and ACO-Lévy flight”. In: *2020 IEEE VLSI device circuit and system (VLSI DCS)*. IEEE. 2020, pp. 378–383.

[22] Saifullah Khan et al. “An Ant-Hocnet Routing Protocol Based on Optimized Fuzzy Logic for Swarm of UAVs in FANET”. In: *Wireless Communications and Mobile Computing* 2022.1 (2022), p. 6783777.

[23] Amgad Madkour et al. “A survey of shortest-path algorithms”. In: *arXiv preprint arXiv:1705.02044* (2017).

[24] Andrew G Barto and Thomas G Dietterich. “Reinforcement learning and its relationship to supervised learning”. In: *Handbook of learning and approximate dynamic programming* 10 (2004), p. 9780470544785.

[25] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[26] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8 (1992), pp. 279–292.

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[28] Haechan Jeong. “Anti-Jamming Path Planning Using GCN for Multi-UAV”. In: *arXiv preprint arXiv:2405.00689* (2024).

[29] Jinlin Peng et al. “Anti-jamming communications in UAV swarms: A reinforcement learning approach”. In: *IEEE Access* 7 (2019), pp. 180532–180543.

[30] Liang Xiao et al. “Two-dimensional antijamming mobile communication based on reinforcement learning”. In: *IEEE Transactions on Vehicular Technology* 67.10 (2018), pp. 9499–9512.

[31] Xin Liu et al. “A heterogeneous information fusion deep reinforcement learning for intelligent frequency selection of HF communication”. In: *China communications* 15.9 (2018), pp. 73–84.

[32] Feten Slimeni et al. “Jamming mitigation in cognitive radio networks using a modified Q-learning algorithm”. In: *2015 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE. 2015, pp. 1–7.

[33] Liyang Zhang, Zhangyu Guan, and Tommaso Melodia. “United against the enemy: Anti-jamming based on cross-layer cooperation in wireless networks”. In: *IEEE Transactions on Wireless Communications* 15.8 (2016), pp. 5733–5747.

[34] Chunyan Jiang, Jingfang Fu, and Weiyan Liu. “Research on vehicle routing planning based on adaptive ant colony and particle swarm optimization algorithm”. In: *International Journal of Intelligent Transportation Systems Research* 19.1 (2021), pp. 83–91.

[35] Lisong Wang et al. “An Energy-Balanced Path Planning Algorithm for Multiple Ferrying UAVs Based on GA”. In: *International Journal of Aerospace Engineering* 2020.1 (2020), p. 3516149.

[36] Y Volkan Pehlivanoglu and Perihan Pehlivanoglu. “An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems”. In: *Applied Soft Computing* 112 (2021), p. 107796.

[37] José Manuel Inclán Alonso and Manuel Sierra Pérez. “Phased array for UAV communications at 5.5 GHz”. In: *IEEE Antennas and Wireless Propagation Letters* 14 (2014), pp. 771–774.

[38] Hong-Ning Dai et al. “An overview of using directional antennas in wireless networks”. In: *International journal of communication systems* 26.4 (2013), pp. 413–448.

[39] Cheol Ung Lee et al. “Tilted-beam switched array antenna for UAV mounted radar applications with 360° coverage”. In: *Electronics* 8.11 (2019), p. 1240.

[40] Roohollah Amiri et al. “A machine learning approach for power allocation in HetNets considering QoS”. In: *2018 IEEE international conference on communications (ICC)*. IEEE. 2018, pp. 1–7.