Insper

Search of shipwrecked people using drone swarms (part 2)

Jorás Custódio Campos de Oliveira (jorascco@al.insper.edu.br)
Pedro Henrique Britto Aragão Andrade (pedroa3@al.insper.edu.br)
Renato Laffranchi Falcão (renatolf1@al.insper.edu.br)
Ricardo Ribeiro Rodrigues (ricardorr7@al.insper.edu.br)

Capstone Project Report
Final Version

São Paulo – SP – Brazil

MAY 2024
Jorás Custódio Campos de Oliveira
Pedro Henrique Britto Aragão Andrade
Renato Laffranchi Falcão
Ricardo Ribeiro Rodrigues

Search of shipwrecked people using drone swarms (part 2)

Capstone Project Report
Final Version

Trabalho de Conclusão de Curso

Undergraduate Capstone Project Report submitted to the respective (Computer/Mechanical/Mechatronic) Engineering programs or Computer Science program in partial fulfillment of the requirements for the Bachelor's Degree.

Advisor: Prof. Dr. Fabrício Jailson Barth

Mentor: Dr. José Fernando Basso Brancalion

General Coordinator TCC/PFE: Prof. Dr. Luciano Pereira Soares

São Paulo – SP – Brazil
MAY 2024
Jorás Custódio Campos de Oliveira
Pedro Henrique Britto Aragão Andrade
Renato Laffranchi Falcão
Ricardo Ribeiro Rodrigues

Search of shipwrecked people using drone swarms (part 2)

Undergraduate Capstone Project Report submitted to the respective (Computer/Mechanical/Mechatronic) Engineering programs or Computer Science program in partial fulfillment of the requirements for the Bachelor's Degree.

Advisor: Prof. Dr. Fabrício Jailson Barth

Capstone Evaluation Committee

_____

Fabrício Jailson Barth

Insper

_____

Fabio Jose Ayres

Insper

_____

Maciel Calebe Vidal

Insper

ABSTRACT

The project's purpose is to iterate on the given multi-agent Drone Swarm Search Environment (DSSE) and research into Reinforcement Learning methods. The DSSE was created with the direct purpose of using reinforcement learning algorithms to train swarms of drones to execute autonomous maritime search and rescue missions of shipwrecked people in the ocean. The environment simulates the movement of persons-in-water (PIW) considering the ocean's dynamic circumstances and calculates a dynamic map of probabilities to be given to the agents, with two distinct environments, one for rescue scenarios with simulated PIW and a second expanding on state-of-the-art research for maritime coverage search path planning. The DSSE facilitates the training and visualization of drone behavior, the project emphasizes continuous improvement and open accessibility, with the release of the DSSE as an open-source Python package and documentation. The focus is on the continuous improvement of simulation quality and applicability of the environments for research purposes, with development, training and evaluation of Reinforcement learning algorithms to improve the path planning of autonomous agents, for search and rescue maritime scenarios.

Keywords: Reinforcement Learning; Shipwrecked people; Drone swarms; maritime search and rescue of persons-in-water; multi-Agent.

1. Introduction

The issue of missing persons-in-water (PIW) is as old as humankind itself, and given the chaotic nature of the ocean, search and rescue (SAR) operations have never been optimal, with limitations on the human ability ranging from the creation of proper search paths to visibility and recognition of PIW.

With all this in mind the DSSE project aims to give a modern solution to the problem. The usage of drones allows for continuous search over extended periods of time and distances, while the capabilities of artificial intelligence (AI) with reinforcement learning for problem solving is still in its infancy, it is theorized that the introduction of AI for SAR applications may significantly improve the efficacy of search operations and reduce the time needed to find and save PIW. It is believed that Reinforcement Learning (RL) can enable the development of new, more efficient search patterns tailored to specific applications. This is based on the hypothesis that the maximization of reward is sufficient to foster generalization abilities, thereby creating powerful agents [1]. Such advancements could potentially lead to the saving of more lives.

Because of the capstone project model, this endeavor is made with a representative of a Brazilian company acting as a client, for both parts of the DSSE project the client represents Embraer, a renowned Brazilian aerospace corporation, which has established itself as a leading manufacturer in commercial and executive aviation, agricultural aviation, and the defense and security sectors since 1969. With a global presence spanning the Americas, Africa, Asia, and Europe, Embraer sets industry standards for innovation and reliability.

This report focuses on building upon the previous capstone project, "Search of Shipwrecked People using Drone Swarms" [2] [3], focusing on refining the Search and Rescue (SAR) environment, and implementing other search algorithms. This project is driven by dual objectives within the SAR training environment: to evaluate and compare various RL algorithms, and simultaneously, the project aims to enhance realism of the training environment by incorporating real-time dynamics, adjusting probability matrices, introducing Probability of Detection (POD), and employing a particle simulation model. It does not address PIW recognition or drone construction, focusing solely on simulation environment development and RL algorithm exploration.

1.1. Project Scope

This project is a continuation of an earlier capstone project “Search of Shipwrecked People using Drone Swarms” [2] [3]. The primary objectives of this continuation are twofold: first, to enhance the training environment, making it more realistic and robust by using real life data, such as drone speed and oceanographic data for current patterns and PIW drift, modeling the drone’s sensor accuracy and validating with a review from an expert in the field; second, to apply and compare various reinforcement learning (RL) algorithms, such as Deep Q-Networks (DQN) [4], Proximal Policy Optimization (PPO) [5]. This approach aims to evaluate the efficacy of these technologies in solving the path planning challenges in Search and Rescue (SAR) missions.

To sustain the transference of results into real world scenarios, a list of improvements to the training environment has been made, focusing on enhancing accuracy, reliability, and applicability. These improvements include:

- Incorporating different movement speeds for the agents (drones) and targets, recreating real world differences between the two.
- Adding intercardinal movements to reflect state-of-the-art maritime SAR simulations.
- A revision of the person's movement and the probability matrix, to better align with real-world research and more accurately reflect realistic conditions.
- Probability of detection (POD) to better replicate the uncertainties of real-life SAR operations. The POD introduces a probability of not detecting the person, influenced by the weather conditions, drone altitude, and other statistics.
- The ability to add multiple PIW at the same time.
- Usage of a Lagrangian particle model [6] to replicate the complex movements and drift patterns of the ocean and model the trajectories of PIW more accurately.
- Inclusion of a second environment expanding on state-of-the-art research for full coverage path planning.
- Deployment of documentation website, aligning it with good open-source practices.
- Automated testing and deployment through GitHub actions.
- Remove collisions between drones and changes so they cannot leave the map.
- Restructure the environment architecture to fit the PettingZoo [7] interface.
- Inclusion of a pre-render time to replicate real world cases where rescue teams require time before arriving at the disaster area.

- Anchoring of the simulation time-step with real time, to facilitate research going forward.
- Rework of the reward function for the DSSE.

To analyze the performance of different algorithms in a simulated environment, considering metrics such as the number of drones and grid size, the following improvements and modifications have been made to the algorithms and agent behaviors:

- Writing of an informed baseline algorithm to have a fairer comparison to the RL techniques.
- Training of RL algorithms with different environment configurations to observe and analyze how the agents behave compared to the baseline algorithm.
- Experiments utilizing a single neural network for the entire drone swarm, and a single neural network per agent.
- Usage of new RL methods: Deep Q-Networks (DQN) [4] and Proximal Policy Optimization (PPO) [5].
- Establishing mechanisms for the exchange of information regarding areas already surveyed by the drones.
- Formulation and testing of specific hypotheses, designed to expand on state-of-the-art research.

The report will encompass all the changes made to the environment, the RL algorithms, their training, and results, with an analysis to determine the efficacy of RL techniques compared to known search patterns and algorithms.

This project does not discuss the recognition of PIW, nor the construction of drones for SAR applications, only the simulation environment and the RL models.

1.2. Resources

The main resource required for this project is the computational power necessary for training Deep Reinforcement Learning (DRL) agents. Insper provides access to a virtual machine hosted on its supercomputer, featuring 12 cores, 128Gb of RAM as well as a NVIDIA Tesla V100 Graphics Processing Unit (GPU) with 32 GB of VRAM. This configuration is believed sufficient for all experimentation related to training RL algorithms.

1.3. Project Timeline

The following table outlines the project timeline from February to June, illustrating key tasks and milestones to be carried out during each month.

Table 1: Activity Schedule for the Reinforcement Learning Research Project.

Tasks	Months				
	February	March	April	May	June
Literature review	x				
Review of the components from the first part of the project	x				
Improvements in simulation environment	x	x	x	x	
Study of Reinforcement Learning models			x	x	
IEEE Fusion Paper	x	x			
Intermediate report			x		
Academic Paper to Journal of Open Source Software (JOSS) [8]			x	x	
Final report				x	
Presentation of project outcomes to professors and executives				x	x

1.4. Stakeholders

Below is a table that maps the main stakeholders of the project.

Table 2: Stakeholder Expectations: Embraer Project Overview.

Stakeholder	Position	Function	Expectations
José Fernando Basso Brancalion	Product Development Engineer - Embraer	Guide the requirements and expectations of the project.	More realistic simulation environment and comparison of search methods in the given environment.
Fabrício Jailson Barth	Professor at Insper and project advisor	Guide the general development of the project.	Delivering the client's expectations on time using a good methodology.

1.5. Risks

During the development of this project, a couple of risks were encountered that could potentially impact its success.

Initially, it was identified that much of the relevant literature and data, such as [9] and [6], were derived from the South China Sea and the Baltic Sea. This regional concentration raised concerns about potential biases in projecting this data onto the training and evaluation of RL algorithms. To address this challenge, the team consulted with the developers of the OpenDrift framework [6], who provided guidance on adapting their simulation for use at any arbitrary latitude and longitude. This initiative effectively mitigated the risk.

Secondly, the training of RL algorithms was expected to demand substantial computational resources and time, potentially making the process unfeasible. To address this issue, the team reached out to Insper to secure a virtual machine capable of supporting the training requirements. This forward-thinking action successfully minimized the risk.

1.6. Ethical and Professional Considerations

As previously stated in the Risks section, the ethical concern of this research lies in possible biasing of trained algorithms due to the data available for ocean water and currents movement. After the guidance from the OpenDrift team, the team was able adapt the environment to use data from any oceanographic region, thus resolving the ethical concerns.

1.7. Standards

The development of this project is governed by the MIT License and adheres to Free Libre Open-Source Software (FLOSS) standards, enhancing the efficiency, safety, and innovation of the software. These standards ensure the software is adaptable, allowing users and researchers to modify it according to their specific needs, thereby making it a user-driven tool.

In addition to utilizing open-source standards, this project employs Git with GitHub, utilizing a version control system establishing a transparent communication channel and effective issue tracking. This setup not only facilitates coordination among contributors worldwide but also enhances project transparency and collaborative input.

Furthermore, the decision to adopt a distributed development model underscores the project's commitment to the principles of open-source collaboration. This model enables global participation and simplifies adaptability, making it easier for diverse teams to contribute to and evolve the project. This approach ensures that the software remains relevant and responsive to the community it serves, fostering a dynamic and inclusive development environment.

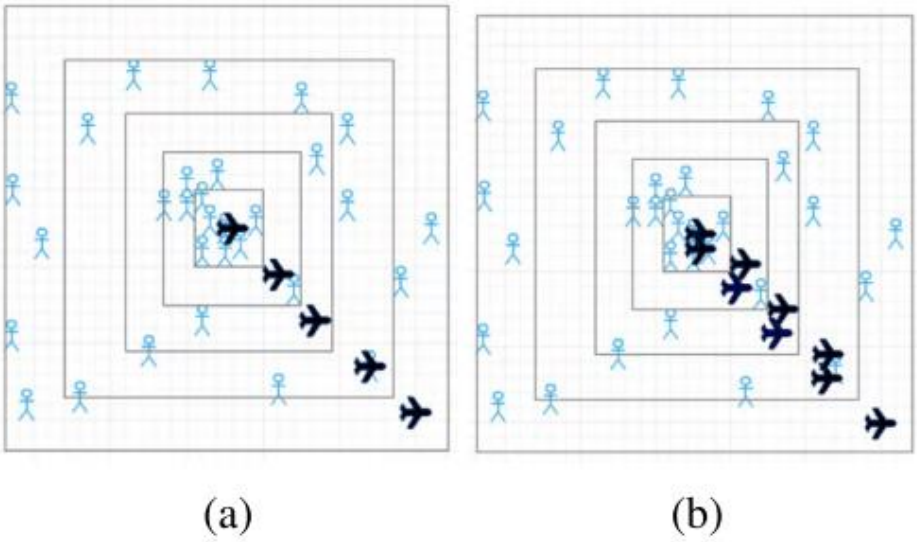
Additionally, by aligning with PettingZoo [7] environment standards, the project can seamlessly integrate with established reinforcement learning libraries. This compatibility greatly facilitates the testing of different algorithms, enhancing the project's utility and the ease with which researchers can explore and implement various approaches.

1.8. State-of-the-Art Review

To enhance the effectiveness of Unmanned Aerial Vehicles (UAVs) in Search and Rescue (SAR) missions, Alotaibi [10] introduces the Layered SAR (LSAR) algorithm. This approach is founded on the premise that in disaster scenarios, there is a central location where most survivors are concentrated. Thus, the algorithm initially concentrates SAR efforts on this focal point before gradually expanding the search area. LSAR leverages advancements in cloud robotics, utilizing internet connectivity to facilitate centralized communication among agents. Under this framework, UAVs communicate with a cloud manager, who orchestrates the SAR operation. Upon receiving information regarding the disaster center, the cloud server running the LSAR algorithm partitions the disaster area into incremental, numbered, square-shaped layers (L). Each layer (Lx) maintains a list of survivors (Lx [SurvivorList]), recording their locations as latitude and longitude coordinates during search missions. These layer lists are managed and updated on the cloud server. Furthermore, the LSAR algorithm employs a fleet of UAVs dispatched by the cloud server to search for and rescue missing survivors. Typically,

each layer is assigned one UAV; however, in certain scenarios, multiple UAVs may be assigned to a single layer to leverage the collective capabilities of multiple agents, the partitioning process can be visualized in Figure 1.

Figure 1: Allocation of drones thought layers using LSAR.



Source 1: "LSAR: Multi-UAV Collaboration for Search and Rescue Missions" [11].

Another study exploring the application of UAVs in SAR missions is presented in "Maritime Search and Rescue via Multiple Coordinated UAS" [12]. This paper introduces a focused unmanned aerial system (UAS) strategy tailored for search operations. The approach involves the coordination of multiple UASs, beginning with the selection and partitioning of the search area based on a probability distribution. Each drone is then assigned a specific partition while maintaining a minimum distance between them. Additionally, a greedy algorithm is employed to match the most capable UASs with the most significant partitions, with capability assessed based on endurance and camera quality, and partition significance determined by survivor likelihood. Subsequent steps involve path evaluation, where each drone is assigned a path and altitude from four predefined options. Furthermore, the paper introduces an equation to calculate the visible area of the drone at any given time.

Now looking at reinforcement learning (RL) solutions, Jia [13] presents a solution to SAR missions in the maritime environment using boats to perform the search, discussing the decision-making problems in SAR missions, which involve determining the search area, deploying maritime vehicles, and planning the search. The authors define metrics to quantify some aspects of SAR missions such as the probability of detection (POD), that introduces the

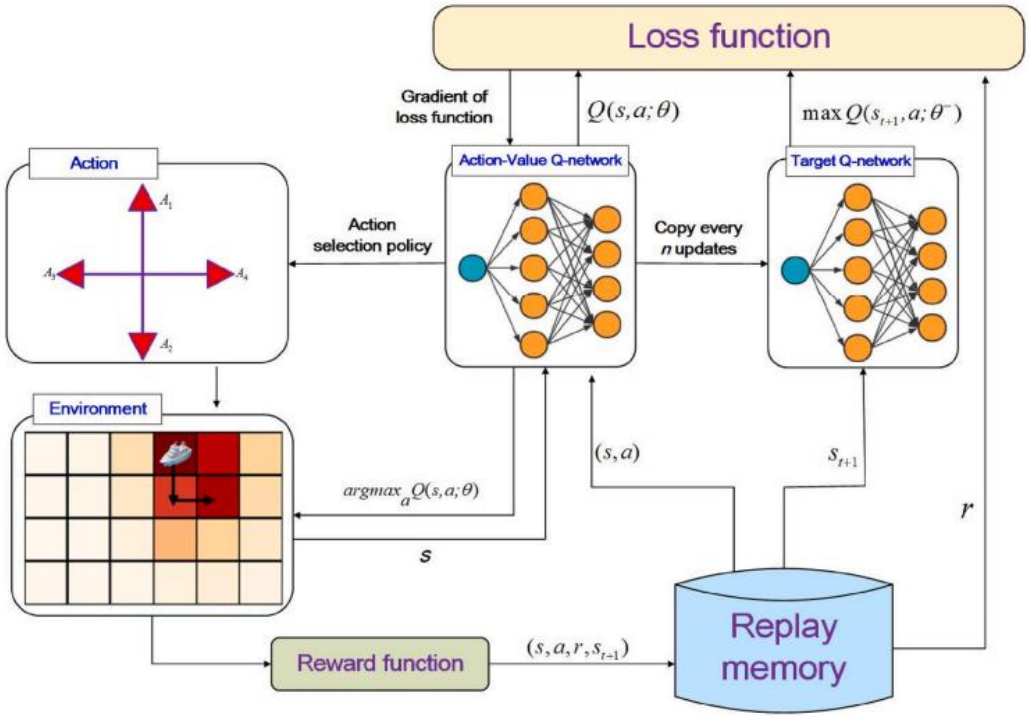
idea of the search vessel to not always find the target in adverse conditions in the maritime environment, as well as the probability of containment (POC), that goes in the same direction of the probability matrix [12], indicating areas that are more likely to contain survivors, and merging both metrics, the probability of success (POS) incorporates both of the metrics to calculate the chance of finding a survivor on the area that is being searched.

Before initiating the search path planning algorithm, this approach incorporates a drift prediction software component. This software utilizes marine dynamic environment data to forecast the trajectory of objects in distress. In the simulation, the distressed object is represented as a particle, and the software predicts the drift trajectory of these particles. Subsequently, based on the projected final positions of the particles, the search grid is constructed. Each grid cell is assigned a Probability of Containment (POC), determined by the number of simulated particles it contains and its distance from the initial point of the drift simulation.

The authors use a classical RL algorithm, the Q-learning, where the next action of the agent is based on the current state. Thus, with the current state as an input, the action that maximizes the POS from its experience (trained Q-table) is selected. The experimental findings demonstrate that the search path generated by the model effectively covers the entire SAR area while minimizing redundant coverage and prioritizing high-probability areas.

Another paper that goes on a similar direction is “An autonomous coverage path planning algorithm for maritime search and rescue of persons-in-water based on deep reinforcement learning” [9], that works with the same metrics, but instead of using classic machine learning, the experiments use Deep Reinforcement Learning (Deep RL), proposing the use of the DQN algorithm, the authors defend that the choice of Deep RL is justified by its ability to combine low-level features into abstract high-level features, approximate nonlinear functions effectively, and excel in perception [14]. Therefore, using Deep neural networks as function approximators, enhances the performance of reinforcement learning algorithms. In this approach, instead of building a Q-Table like in the previous [13], the algorithm uses Deep neural networks to approximate the values, and thus choosing the best action given the current state of the environment. The selection of agent action for this algorithm is like the Q-learning algorithm, the difference being the fact that instead of an in-memory Q-table to store experiences, it uses a neural network, a diagram to illustrate the functioning of the algorithm can be seen in Figure 2.

Figure 2: Action selection and training diagram for DQN.



Source 2: An autonomous coverage path planning algorithm for maritime search and rescue of persons-in-water based on deep reinforcement learning [10].

In their experiments, they found that the proposed algorithm had results that at least match the q-learning approach results or even take less time to complete the SAR mission. However, both studies have some limitations, such as the fact that the search area is assumed to remain constant during the path planning and thus, it does not consider aspects like wind, waves and moving obstacles during the search.

Both the LSAR [10], the Q-learning [13], the UAS [12] and DQN [9] approach utilize probability regions to locate targets. However, while the LSAR approach consistently centers the highest probability region, [10] [13] determines probabilities based on maritime data and statistical methods. Furthermore, UAS [12] and LSAR [10] employ multiple agents to expedite SAR missions. In LSAR [10], drones communicate only upon individual detection, whereas in UAS [12], agents operate independently without inter-agent communication. Despite their shared objective of optimizing SAR missions and enhancing success rates, each approach employs distinct strategies to achieve these goals.

2. Methodology

The choice was made for Agile methodologies, a dynamic and collaborative approach created in the early 2000s. The Agile approach was developed to introduce a more dynamic and collaborative framework in project management. Agile was born from a need to transcend the rigid and cumbersome processes that dominated project management, advocating for adaptability, continuous improvement, and team collaboration. Its emphasis on flexibility seeks to foster team synergy and boost productivity [15]. This paradigm shift not only facilitates more efficient project development but also ensures alignment with the evolving requirements of the contemporary digital landscape, guaranteeing that outputs are not only of superior quality but also pertinent and timely.

Within the Agile framework, three methodologies stand out due to their widespread adoption: Scrum, Extreme Programming (XP), and Kanban. The project team has opted for the implementation of Kanban, recognizing its ability to enhance productivity by minimizing time spent on non-essential communication. Kanban is distinguished by its visual task management system that usually consists of three columns, representing distinct phases of the development process: “To Do”, “In Progress”, and “Done”. Each task is represented by a card that moves from one column to the next, reflecting its current stage, providing a transparent overview of work in progress, thereby facilitating more accurate time estimates and task prioritization. This methodology's principle of limiting work in progress directly contributes to a clearer delineation of team members' responsibilities.

Standing as one of the most recognized and utilized code hosting platforms globally. GitHub's adaptation of the Kanban system facilitates a logical integration of software development with Agile project management. The platform allows us to organize tasks efficiently into columns such as “To Do”, “In Progress”, and “Done” which are standard. Additionally, it was decided to incorporate three new columns titled "Validation Needed", "Backlog", and "In Test" into the project management framework. The "Validation Needed" column is designated for ideas that necessitate collaborative discussion among the group, the advisor, and the client. The "Backlog" column serves to compile all tasks validated for the project, while the "To Do" column lists tasks scheduled for completion within the sprint. Finally, the "In Test" column is reserved for tasks that have been completed and require approval from the rest of the team and the stakeholders. With the chosen methodology, confirmation of progress was consistently received during the weekly meetings with the advisor. Furthermore, validation of ideas and reporting of progress were conducted biweekly

with the client until one month before the deadline, after which these meetings started to occur weekly.

The table below represents the team's progress through various sprints:

Table 3: Agile Sprint Overview: Embraer Project Development Phases.

Sprints	Data Start	Data End	Results
1	29/01/2024	06/02/202	Introduction and understanding of the project according to the client.
2	07/02/2024	14/02/202	Refactored the code for the first part of this project, fixed the bugs that were identified, updated the Python package to integrate with the requirements, and resolved some bugs in it.
3	14/02/2024	21/02/202	Added four more movements to the drones (in the diagonals), implemented automated testing with GitHub Actions and Pytest, and conducted more in-depth research on papers about simulating shipwrecks people. Implementation of the Greedy algorithm.
4	21/02/2024	29/02/2024	Integrated the POD into the simulation, updated the RL model for the new experiments, began the paper writing process for IEEE Fusion, and conducted data collection for it.
5	29/02/2024	06/03/2024	Identified and resolved inefficiencies in the probability matrix calculation, avoiding the recalculation of some parts and using optimization tools. Implemented a real time based timestep in the simulation.
6	06/03/2024	14/03/2024	Conducted comparisons between algorithms with a shared neural network among the drones and an individual neural network for each drone. Finished the IEEE Fusion Paper.
7	14/03/2024	21/03/2024	Integrated the pre-render simulation with the search environment.

8	21/03/2024	29/03/2024	Modified the environment to keep running when the drone exits the map boundaries, while maintaining the negative reward.
9	29/03/2024	06/04/2024	Researched the Lagrangian particle model, scheduled a meeting with the previous group that worked on this project, and changed the environment so that it is possible to add more than one PIW.
10	06/04/2024	14/04/2024	Create a variable to modify the POD for each PIW, developed more tests for the environment, adapted the environment to the PettingZoo interface, reviewed the Gaussian distribution calculation on the probability matrix and improved the PIW movement.
11	14/04/2024	21/04/2024	New neural network architecture for experiments, continue building the coverage environment, created a VitePress website to host the documentation, added the environment to the official PettingZoo third party environments list, added tests for the Coverage Environment, and made the project more welcoming for new contributions.
12	21/04/2024	29/04/2024	Finished the Coverage Environment, completed the documentation for the Search Environment and Coverage Environment, and submitted a paper to JOSS [8].
13	29/04/2024	06/05/2024	Training of RL algorithms, data gathering and comparison analysis.
14	06/05/2024	14/05/2024	Writing of the final report, review, and submission.

2.1. Environment

For the experimental phase of the project, the DSSE framework, available as a Python package [16], was used to train and validate algorithms to study the viability of using reinforcement learning in search of shipwrecked people. The environment, as an open-source project, received contributions from the team to enhance its capabilities, efficiency, code readability and best practices. The main purpose is to create a more realistic environment and a more receptive ecosystem to future contributions.

Guided by the principles outlined by Robert C. Martin and Michael C. Feathers in "Clean Code" [17], the team refactored the codebase to enhance readability, maintainability, and performance. Also, the movement of the drones was criticized by experts during a review of the first part of the project, in which the absence of diagonal movement was highlighted as a limitation, arguing that this feature would enhance the realism of the simulation. To address this issue, diagonal movement was prioritized and introduced on the agents to enhance its navigation capabilities.

To ensure the robustness of future code changes, automated testing was implemented using GitHub Actions and the Pytest framework, facilitating continuous integration and deployment processes. The tests empirically assess the system's behavior across various scenarios and stimuli, evaluating its ability to handle errors, accurately create the environment, and deliver precise results and data to the user.

During the development of the environment, the team discovered that the existing environments did not adhere to PettingZoo's [7] interface standards. This incompatibility prevented the use of existing RL libraries, necessitating that each algorithm be implemented from scratch. This limitation also made it challenging to use more efficient training solutions, such as the distributed Reinforcement Learning framework RLlib [18]. To address this issue, the team redesigned the software architecture to comply with PettingZoo's recommended interface. Furthermore, to verify proper implementation and adherence to PettingZoo's standards, a specific functionality test provided by PettingZoo was integrated into the project's suite of tests. This architectural revision made the environment compatible with any RL libraries that support multi-agent environments.

Changes were also made to possibilities the overlap of agents on the same cell and to prohibit the agents from leaving the simulation grid. The overlap was added due to the cell size of 130m, a size that easily allows the presence of multiple drones without collisions. While disallowing the agents to leave the grid was done because in SAR missions, there is no reason

allow the agents to leave the search area, and there can be additional layers of software controlling the drones.

To facilitate the development and to properly differentiate intended uses, two distinct environments were established: a “search” environment, featuring PIW, and a "coverage” environment, which does not contain any PIW and only utilizes the probability matrix.

2.1.1. Search Environment

The search environment is the original scenario created by the starting team and was designed for training RL agents to locate PIW. It simulates the movement of PIW in real time, concluding when all PIW are found, or a certain time step limit is reached. This environment incorporates relevant variables to the search scenario, such as the POD, number of PIW, and vector values that guide the movement of both the PIW and the probability matrix.

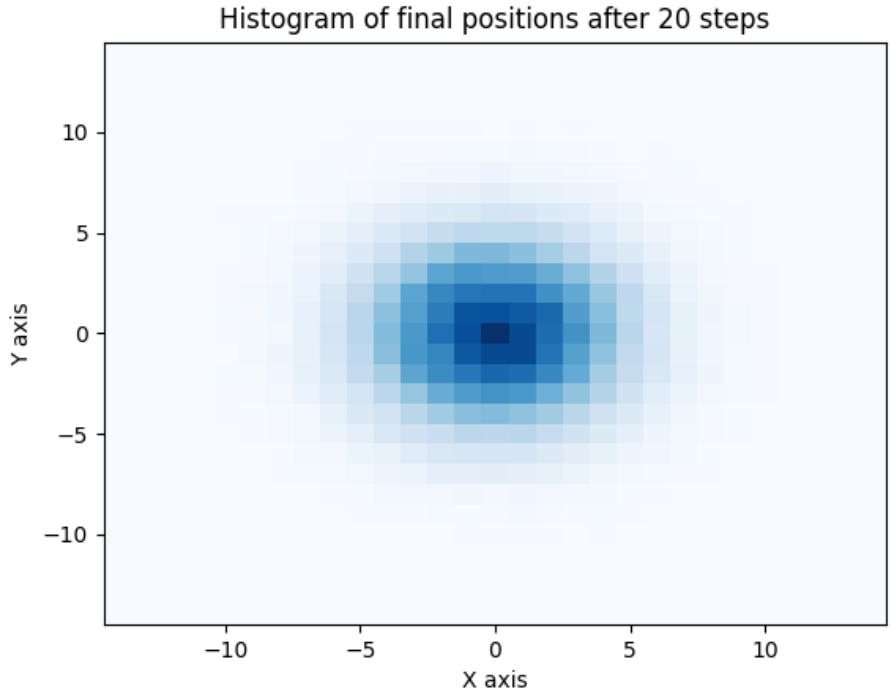
A critical feature, the POD, was integrated into the simulation to mirror the real-world uncertainty of finding shipwrecked individuals in the maritime environment, this addition affects the agents, giving them a fixed probability of finding the target during a search action. This change was made to approximate the environment to real world scenarios where the visibility is hindered by weather factors. Furthermore, the simulation's time step was adjusted to reflect the real-world speeds of drones relative more accurately to the water's drift that carries away individuals, to produce a more realistic environment and simulation times.

Further key features related to the shipwreck survivors were integrated into the environment, including the addition of more survivors and enhanced movement of individuals. The number of survivors can now be set as a parameter during the environment’s creation. Each survivor moves independently, using the probabilities of the surrounding cells to choose a new position. These probabilities are defined by a two-dimensional Gaussian, characterized by a radius that expands over time and a center that shifts dynamically. Both the radius increasing factor and the vector that guides the movement of the gaussian are adjustable parameters of the environment. The idea of the Gaussian function comes from the mathematical concept known as the random walk, which describes the path of successive random steps on a mathematical space. When simulating the path performed by an individual after just 20 of random steps, with equal probability of moving to any adjacent cell or staying on the same cell, the histogram in Figure 3 clearly depicts that the probability of containing the individual of the cells surrounding the origin resembles a two-dimensional Gaussian function.

The project adapts this concept by incorporating a noise-augmented vector. This vector introduces a divergence in degrees from the movement vector, influencing the movement of the

high probability region. By adding this vector to the existing probabilities neighboring the individuals, it slightly increases the probability of moving towards the direction and orientation of the noised vector. This modification ensures that movements are not wholly predictable and allows individuals to occasionally move outside the areas of highest probability.

Figure 3: Visualization of the random walk.



Source 3: Made by the authors.

Another key aspect of the environment that was modified was the rewards structure, it has been changed from a semi-dense reward structure to a sparse one. The environment used to give rewards on the action of walking, on searching on the cells with greater probability, and on finding the person. This has been streamlined so that agents now only receive rewards upon successfully finding the target. Experiments have demonstrated that this sparse approach fosters better-performing policies. A more detailed analysis of these findings will be provided in the results section. Below is the definition of the reward function.

$$R(T_S) = \begin{cases} 2 - \dfrac{T_S}{T_{S_{limit}}}, & if\ PIW\ found \\ 0, & otherwise \end{cases}$$

Equation 1: Environment's Reward Function.

Being, the T_S the current time step, and $T_{S_{limit}}$ the limit of time steps for the current SAR mission.

Performance improvements came from modifications made to the probability matrix, during experiments, the team identified that the interaction with the environment during the training of the neural networks was taking more time than expected, as well as using 100% of the CPU core that the process was launched. Given this issue, an investigation using cProfile [19], Python's default profiler, revealed that the step function of the probability matrix was the most time- and CPU-intensive task. To mitigate this problem, the code was changed to remove unwanted re-calculations in the step function, as well as the incorporation of the numba [20] library, which implements a JIT compiler to optimize python functions using fast machine code. In a tested scenario, where the matrix was iterated five thousand times, the optimized code was approximately 88 times faster, reducing the average time required for each simulation step from 6.665 milliseconds to just 75.028 microseconds. Consequently, in practical terms, the training time for the algorithms decreased by over 10 hours.

2.1.2. Coverage Environment

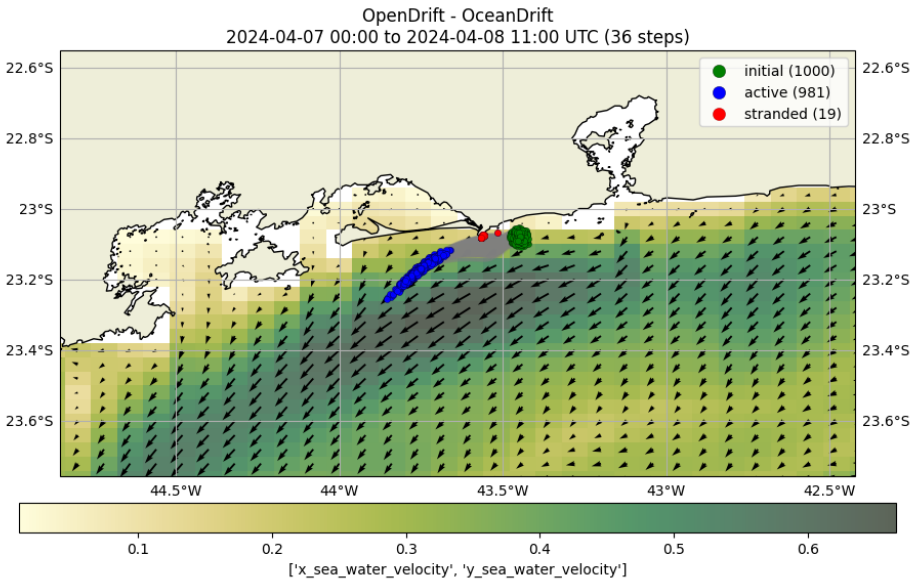
The Coverage environment was designed based on state-of-the-art research in maritime coverage search path planning [9] [13]. This complex field blends coverage path planning with maritime Search and Rescue (SAR) operations, akin to a variant of the traveling salesman problem [13]. Trummel and Weisinger [21] have demonstrated that finding an optimal search path, where the agent must search all sub-areas using the shortest possible path, is NP-complete. Considering this complexity, the environment is specifically tailored to reward agents for maximizing area coverage without redundant searching, while prioritizing areas with a greater Probability of Containment.

Furthermore, this environment excludes shipwrecked individuals, this shift emphasizes the development of strategies that optimize exploration space rather than the rapid location of targets, providing a unique challenge and a valuable tool for studies in autonomous search operations with the Coverage Environment.

This environment includes exclusively variables relevant to the coverage objective and uses a more advanced probability matrix, which integrates a Lagrangian particle model [6] with maritime and wind satellite data to achieve greater precision, the Lagrangian model was excluded from the search environment because of its long processing times required to generate the probability matrix. Implemented by OpenDrift [6], this model treats the PIW as particles, and applies current and wind data, predicting the final position of a PIW based on the initial position and the time of drift, a complete drift trajectory with the maritime currents can be seen on Figure 4. The result of this simulation is a list of latitude and longitude points for each of the

particles, with this it is possible to use some mathematical transformations to create the new probability matrix.

Figure 4: Image illustrating the drift process (35h) on the software.



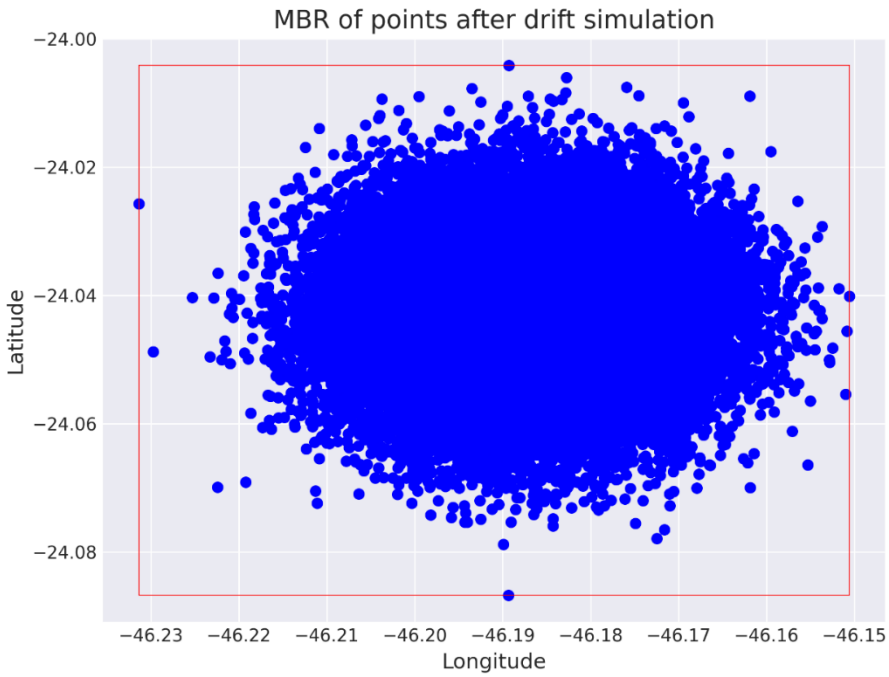
Source 4: Made by the authors.

The first transformation implemented is a minimum bounding rectangle (MBR) method. This method processes a list of points to identify the points with minimum and maximum latitude and longitude. With these points, the haversine formula - expressed in Equation 2, where R is the radius of the earth, φ is the latitude of the points and λ is the longitude – calculates the distance between geographical coordinates in meters on the Earth's surface. This calculation helps determine the search grid size. The grid cell size, a configurable parameter, defaults to 130m. Using the formula, the grid's width and height can be calculated, and the measurements converted into grid size. This process is depicted in the Figure 5.

$$d = 2R \cdot \arcsin\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + cos(\varphi_1) \cdot cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}$$

Equation 2: Haversine formula.

Figure 5: Minimum bounding rectangle of the final positions of the drifted particles.



Source 5: Made by the authors.

Once the map size is calculated, the latitude and longitude coordinates must be converted to x and y coordinates on the simulation grid. This is achieved through an equirectangular projection – expressed in Equation 3 – where R is the radius of the earth, λ is the longitude to project, λ_0 is the central meridian of the plane, φ_1 is a latitude close to the center of the plane, φ is the latitude to project and φ_0 is the central parallel of the plane - which translates the points on the spherical surface of the Earth to the simulation's Cartesian plane. Having each particle's (x, y) position established on the grid, the Probability of Containment (POC) is calculated via a straightforward counting method: the POC is derived from the ratio of particles within a grid cell to the total number of particles. After implementing these transformations, the probability matrix is generated, integrating both simulation and statistical data instead, moving beyond pure mathematical expressions.

$$x = R \cdot (\lambda - \lambda_0) \cdot cos(\varphi_1)$$

$$y = R \cdot (\varphi - \varphi_0)$$

Equation 3: Equirectangular projection.

In this environment, the observations are equal to the search environment, where agents receive a tuple consisting of their position and the probability matrix. The key difference is that in the coverage environment, the probabilities of cells that have already been searched are set to 0. The objective is to cover the whole area minimizing repeated searches and prioritizing

areas with higher probabilities. Therefore, it is important to include information about the searched cells in the observations and provide the complete probability matrix.

This environment incorporates multi-objective optimization, expecting agents to learn to minimize coverage time while prioritizing cells with greater POC simultaneously. The reward structure reflects how agents learn to optimize these tasks. If one task yields a greater discounted sum, agents tend to prioritize that task and treat the other as a sub-goal. This poses a challenge in developing an effective reward function. To address this, the team developed a simple reward function that tries to balance the objectives. The reward function is described in Equation 4.

$$R(T_S) = \begin{cases} R_{done}, & if\ S_{t+1} = S_{done} \\ 1 + R_{poc}, & if\ S_{t+1} \neq S_{done}\ \&\ S_{t+1} \notin V \\ -0.2, & else \end{cases}$$

Equation 4: Reward function for the coverage environment.

Where S_{done} is the state where all the cells with POC greater than zero have been searched, V is the set that holds all the non-searched cells, R_{done} is calculated by Equation 5 and R_{poc} is calculated by Equation 6.

$$R_{done} = n_{cells} + n_{cells} \cdot \left(1 - \frac{T_S}{T_{S_{limit}}}\right)$$

Equation 5: Calculation of the reward on conclusion of the coverage.

$$R_{poc} = POC \cdot n_{cells} \cdot \left(1 - \frac{T_S}{T_{S_{limit}}}\right)$$

Equation 6: Calculation of the POC related reward for the coverage environment.

Where POC is the probability of containment in the searched cell, T_s is the search's timestep, $T_{S_{limit}}$ is the limit of the timesteps for the simulation and n_{cells} is the number of cells with POC greater than zero. The fraction on R_{done} and R_{poc} is used to stimulate the agents to conclude the search as quickly as possible, giving out greater rewards for faster searches, while the usage of the number of cells in the formula is used to keep the discounted sum of the rewards in the same scale, so that all tasks are made relevant to the agents.

2.2. Algorithms

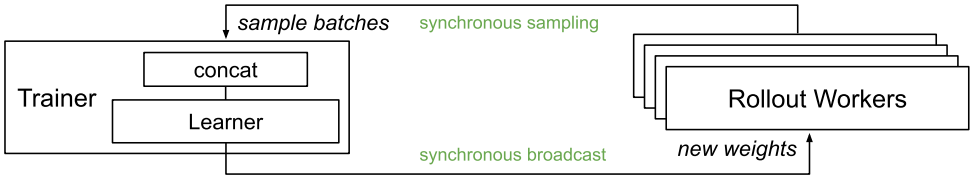
To effectively compare and evaluate multiple search algorithms for the problem, the team implemented from the ground up three key algorithms: a new baseline algorithm, an independent learning version of Reinforce and DQN [4]. Additionally, the team utilized the RLlib [18] library to implement the PPO [5] and another version of DQN, supporting both

centralized and decentralized learning configurations. The reason behind the choice of DQN and PPO is to study the main algorithms from the value-based and policy-based families, respectively.

Value-based and policy-based are two distinct families of Reinforcement learning algorithms, with diverging objectives and fundamental approaches to solving the same problems. The family of value-based algorithms focuses on approximating the value function for the given environment, through the trained Q-Table or Q-Network. Meanwhile, policy-based algorithms focus on approximating an ideal policy for the given environment, with a mapping of states to actions, this is commonly made with neural networks, and thus policy-based algorithms do not directly map all possible actions and states to rewards, but instead attempt to directly optimize the policy's parameters to maximize cumulative rewards [22].

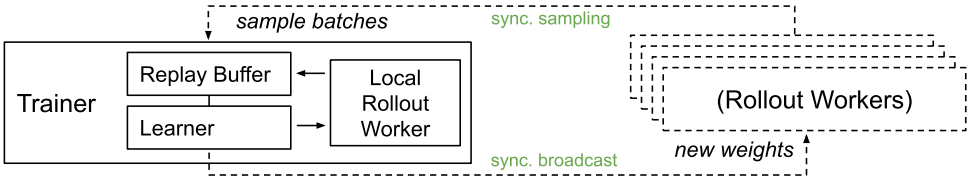
Before delving into specific differences between the two algorithms, as the tests were made using a distributed RL library, RLlib [18], it's important to know the architecture differences that it brings, as shown on Figure 6 and Figure 7, the training process is clustered, making use of multiple Rollout processes (workers) and a centralized Trainer. This setup works by using the Rollout Workers to collect samples (experience) from the environment in parallel, and gathering this information on the trainer, which updates the neural network, and sends the updated version to the Rollout Workers, so that the process can repeat until the end of the training. The main advantage of the usage of the RLlib library, is that the training process is sped up by a lot (a reduction in training time almost equivalent to the number of workers), as the training process used to take 24~48 hours, and with the usage of 11 Rollout Workers (1 per core) plus 1 Trainer worker, the process started taking up to 3~8 hours depending on the experiment.

Figure 6: RLlib's training architecture for PPO.



Source 6: RLlib's documentation [23].

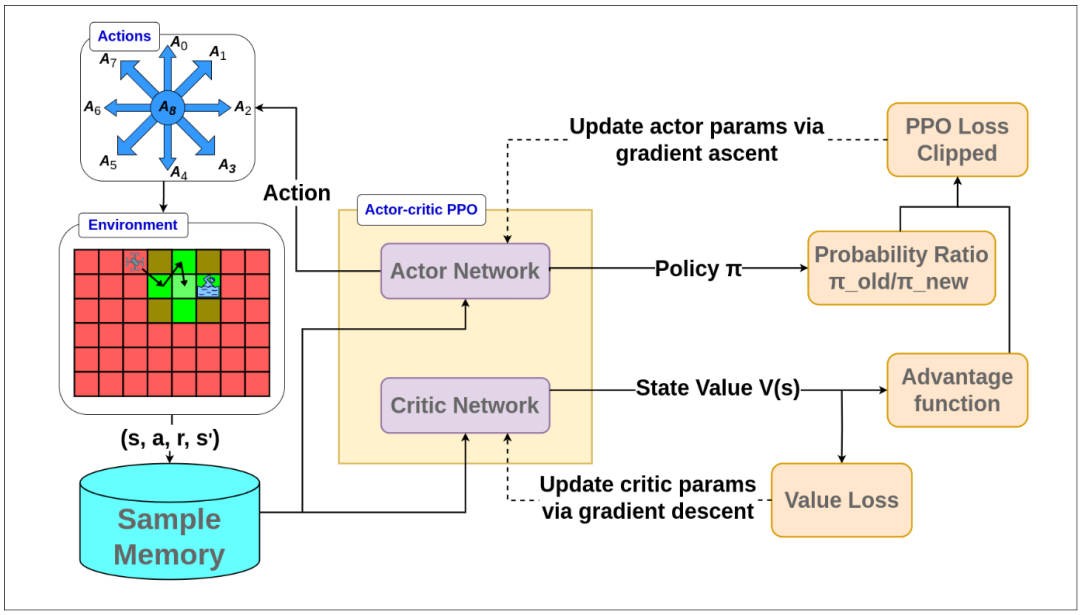
Figure 7: RLlib's training architecture for DQN.



Source 7: RLlib's documentation [23].

As it can be seen in Figure 6 and Figure 7, there are some minor differences in the training implementation of the algorithms in the library, this is due to intrinsic differences in the workings of the algorithms. To better understand the workings of the algorithms, diagrams were formulated illustrating the training process of each of the algorithms, in Figure 8 PPO's training process is illustrated, in this algorithm, there is two main parts, the actor, that is responsible for interacting with the environment, and collecting the experience stored in a sample memory, and the critic, that is responsible for optimizing the policies when the stored experiences collected by the actor reach a certain number (in the case of RLlib, this is defined as the training batch size, that is collected in parallel). Another key feature of PPO is its clipping function, this function limits the gradient between a given range, to ensure that no big parameter update is made in one go, this was proposed to resolve the instability issues of other policy-based algorithms like Reinforce, as well as to mitigate the cases where the policy gets stuck at a local maximum.

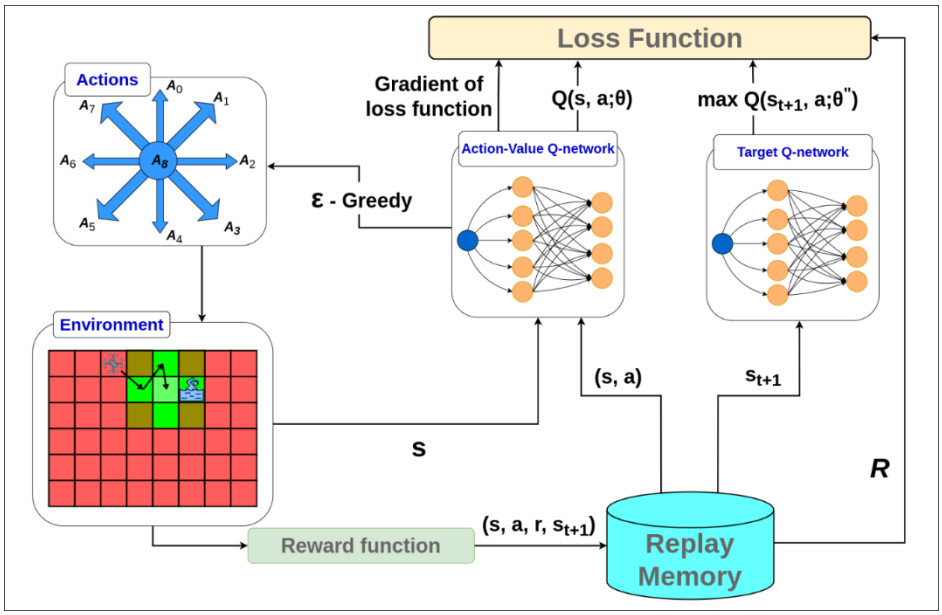
Figure 8: Training Process Diagram for PPO in the developed Environment.



Source 8: Made by the authors.

In the case of DQN, illustrated in Figure 9, we have two networks as well, an Action-Value Q-network and a Target Q-network, the action-value is the neural network that chooses the actions, based on its estimated Q-values, the target network, on the other hand is used to calculate the loss, and update the action-value network, the weights of the Action-Value network are copied to the target network in a defined threshold of steps. While the Action-Value interacts with the environment, it stores its experiences in the replay memory, in another predetermined threshold, the algorithm samples experiences from the replay memory, and executes what is called a “experience replay” where the estimated Q-values are updated based on the data collected from the experiences (states, actions and rewards), making the policy be indirectly updated.

Figure 9: Training Process Diagram for DQN in the developed Environment.



Source 9: Made by the authors.

As pointed out previously, the two algorithms have differences in the way they learn, but there are some key differences in the way they explore the environment, which is a key factor of the learning process in Reinforcement Learning. The difference in their exploration, reflects from the way each of the algorithms choose their actions, PPO chooses its actions sampling the from the probability distribution of actions, outputted by its policy network, this makes it so it has a higher chance of exploration when the agent is less certain on the optimal state, action mapping and a higher chance of exploitation when the agent has learnt more on the given (state, action) pair. As for DQN's exploration, it chooses its actions based on a $\varepsilon - greedy$ algorithm, that has a chance of sampling a random action of the environment (exploring), and a chance of choosing the action with the highest Q-value (exploitation).

The baseline algorithm does not make use of Reinforcement Learning but uses the probability matrix to calculate the agent's actions. This algorithm was constructed under the simple heuristic of each agent of the drone swarm receiving a cell of the highest probability on the probability matrix, and searching there, being an algorithm that focuses only on exploitation. This algorithm was selected based on the simplicity of its idea and implementation and its non-stochastic behavior, with the intent of using it to compare with the RL approach to better understand its efficacy.

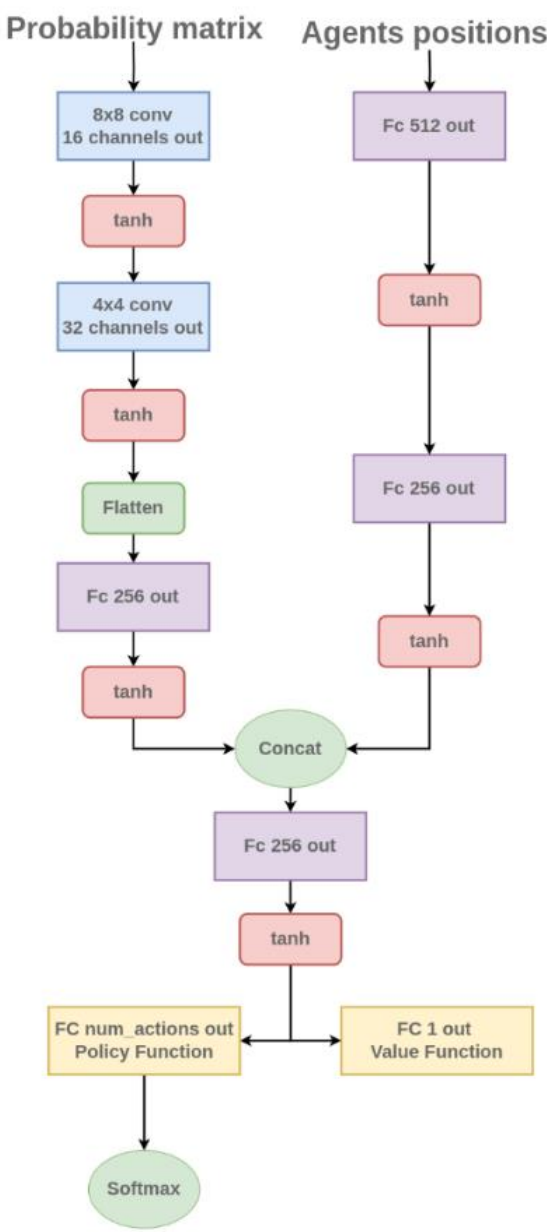
An additional iteration of the previously tested Reinforce algorithm [2] [3] was also proposed. The initial implementation employed a centralized learning approach, where all agents shared a single neural network. Building upon this, an independent learning approach

was introduced, in which each agent operates its own neural network to select actions and learn from its individual experiences. The decentralized version of the algorithm was implemented by hand to achieve this setup. This modification emphasizes the collaborative nature of the environment, fostering inter-agent cooperation and enhancing the effectiveness of the SAR mission.

One of the key factors that was also changed in the algorithms front was the neural network architecture, in the early experiments, the neural network used was a fully connected, that received only the positions of the agents and the positions of the top 10 greatest probabilities, this setup did not make that sense anymore on a situation where the PIWs can move out of the zone with greater probabilities, and the results of some initial experiments using this setup on the updated environment, as expected, were not promising, failing to learn an ideal policy.

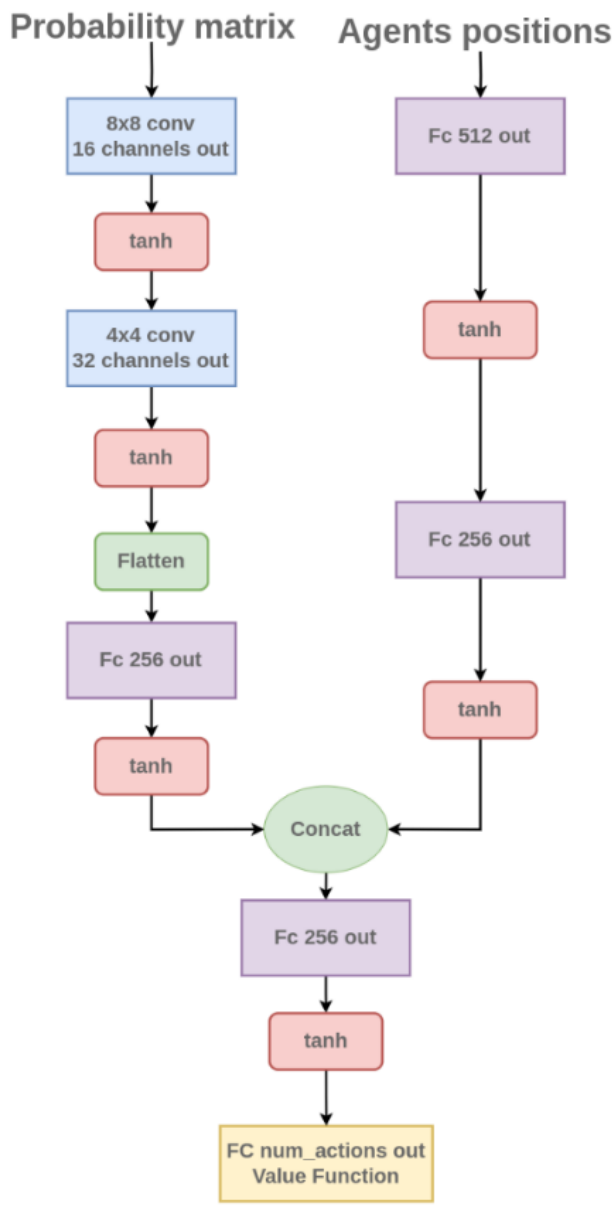
Based on that, the team iterated over some possible architectures, and the one that showed the best results was a Multi-input Convolutional neural network (CNN), where there is a CNN branch that receives the full matrix, and a Fully Connected (FC) branch that receives the agents positions, a more detailed diagram of this setup is on Figure 10 and Figure 11. The CNN branch and the FC branch are concatenated, and flow through a joint FC layer. From there, in PPO there are two outputs, a policy branch, which determines the action probability distribution, and a value branch, used by the critic to update the policies. In DQN, there is only one output branch, which is the Q-values for each of the actions in that state.

Figure 10: Neural Network Diagram for PPO Implementation.



Source 10: Made by the authors.

Figure 11: Neural Network Diagram for DQN Implementation.



Source 11: Made by the authors.

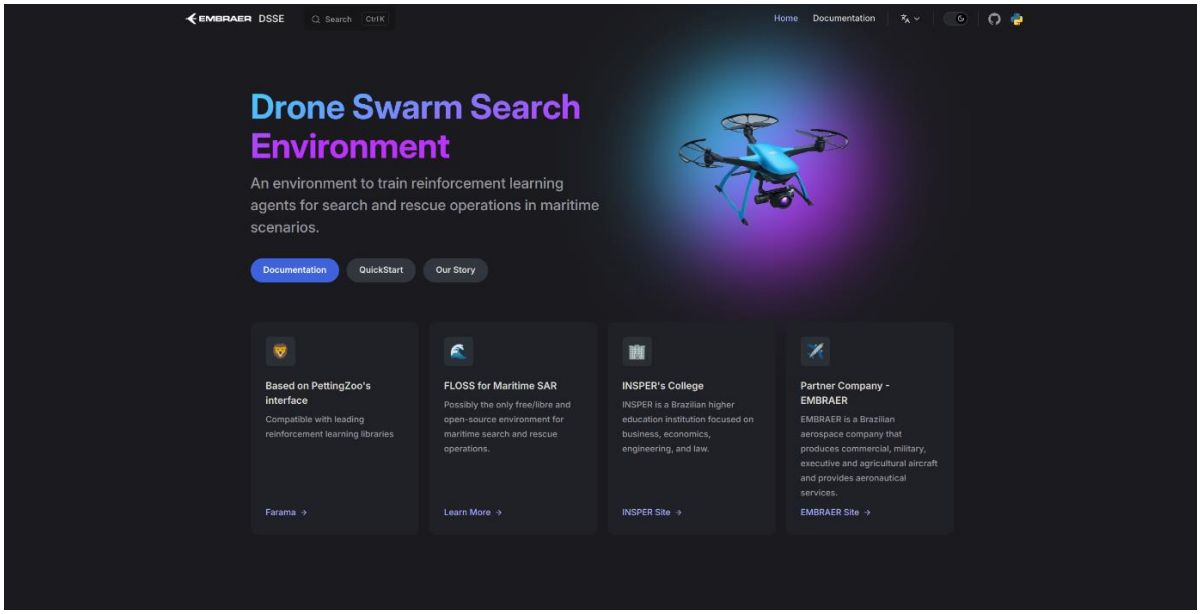
2.3. Documentation

Effective documentation is a cornerstone of any successful open-source project. It not only serves as a guide for current users and developers but also plays a crucial role in attracting new contributors, users, and researchers. By providing clear, comprehensive, and accessible documentation, a project can significantly enhance its visibility and usability. Well-documented projects encourage a broader community to engage and contribute, fostering further advancements and ensuring sustained interest and participation in the field. Recognizing this,

the team decided to invest time and resources in creating thorough documentation for this project, aiming to maximize its impact and facilitate future developments in the area.

In the first part of the project, all documentation was initially maintained within the repository's README file. However, as the project grew, highlighted by improvements in the search environment and the introduction of a new coverage environment, the complexity of the information needed to instruct users increased substantially. Consequently, relying solely on README became insufficient for the expanded scope of the project. Understanding the value of resolute and well-written documentation, the team decided to develop a new, more detailed documentation website using VitePress. This platform not only provides a more organized structure but also offers a visually appealing aesthetic, enhancing the user experience.

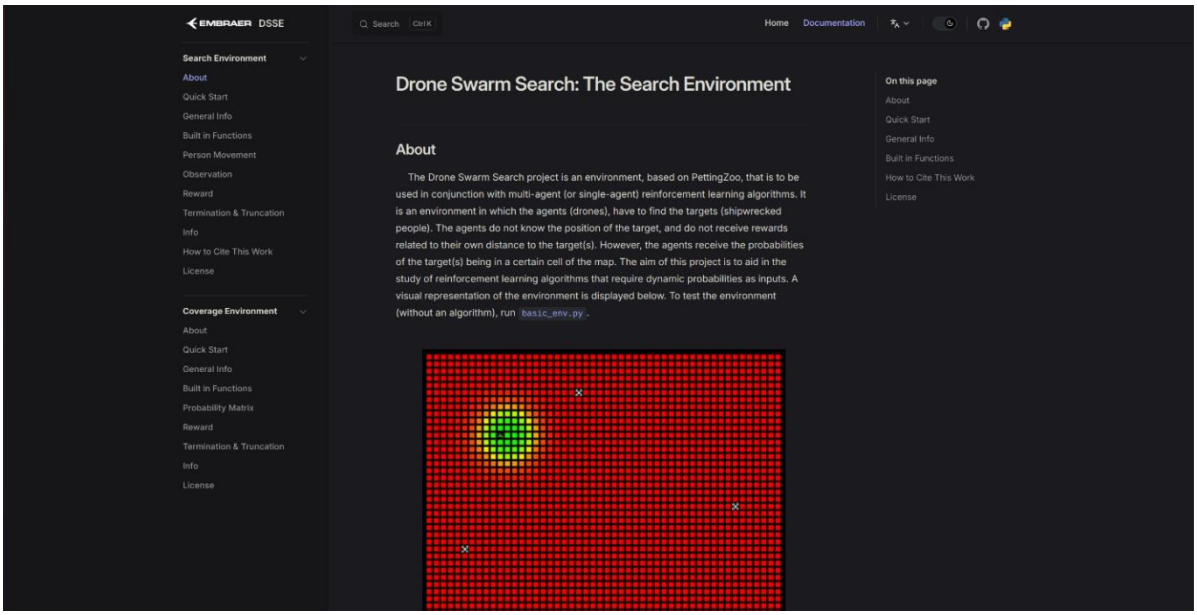
Figure 12: Home page of the Documentation.



Source 12: Made by the authors.

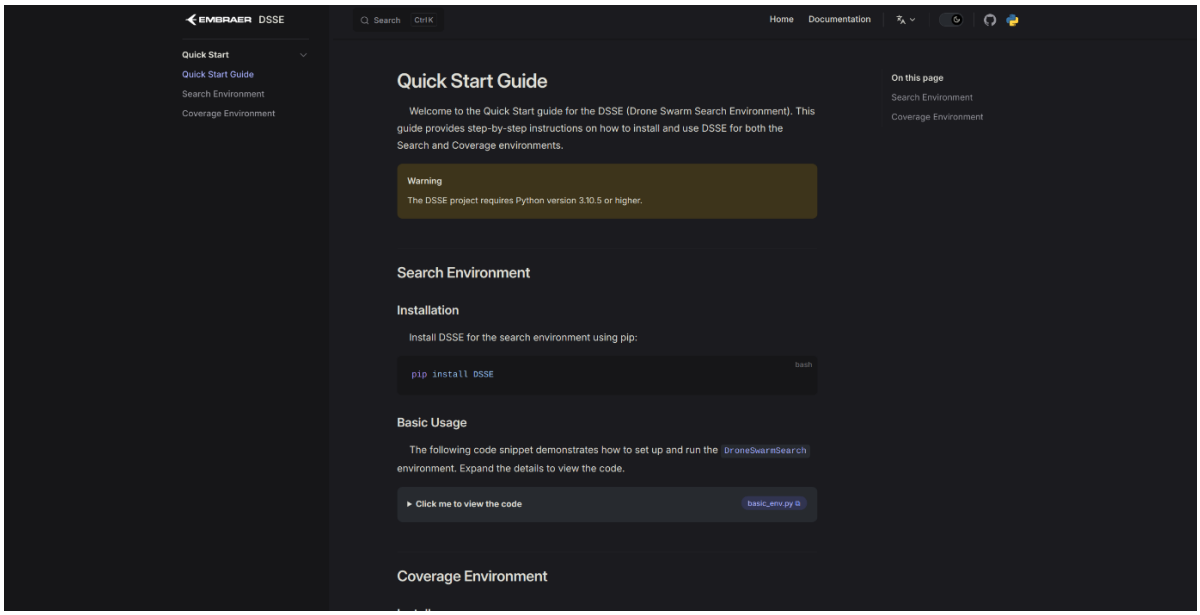
Figure 12 illustrates the main page of our documentation. It features navigational buttons for accessing detailed documentation, the search and coverage environments. A "Quick Start" section provides concise instructions on setting up and using both environments, including installation procedures and example codes. Additionally, there is a button that opens a brief overview of the project's inception.

Figure 13: Documentation page of the site.



Source 13: Made by the authors.

Figure 14: Quick Start page of the Documentation.

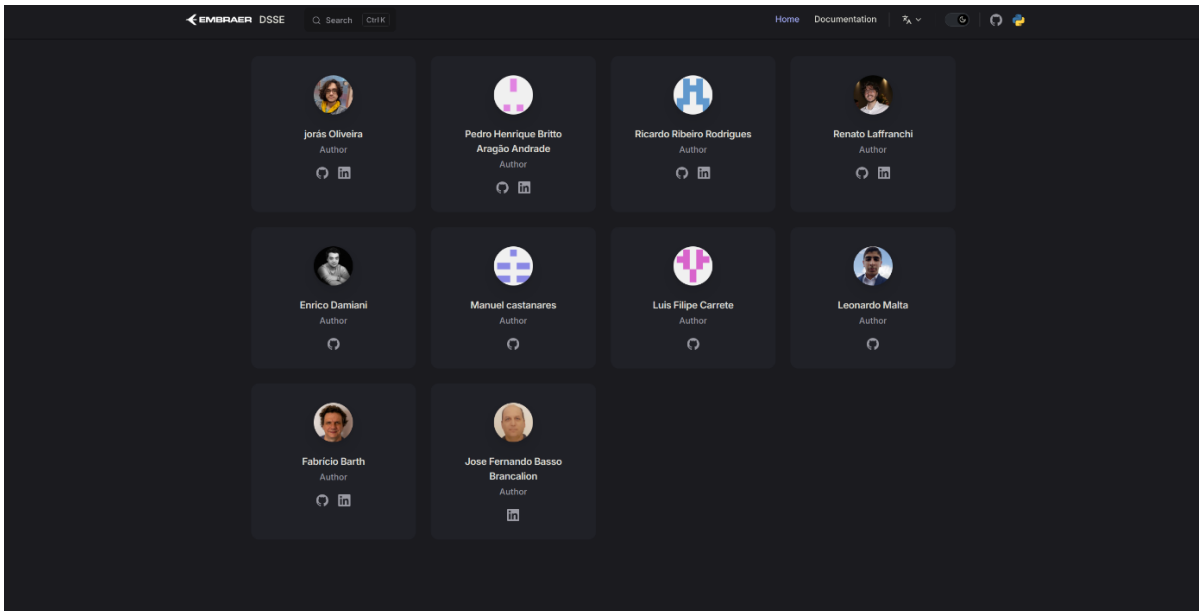


Source 14: Made by the authors.

Furthermore, Figure 12 includes four additional blocks. The first block provides a link to the official site of Farama's PettingZoo, which inspired our interface design. The second block, linking to FLOSS, our project is what we believe to be the pioneering open source initiative for maritime search, although FLOSS does not officially highlight it as such. The third block directs users to INSPER college, the institution supporting this research. Lastly, the fourth block details our collaboration with the partner company, Embraer.

Additionally, the homepage features a section listing all contributors to this project, as shown in Figure 15.

Figure 15: Contributors Section.



Source 15: Made by the authors.

Figure 13 displays the documentation section for the Search and Coverage environment. The layout and content of this section were designed to align with Farama's official documentation. Our objective was to improve accessibility, thereby streamlining the integration and adaptation process for newcomers engaging with the project.

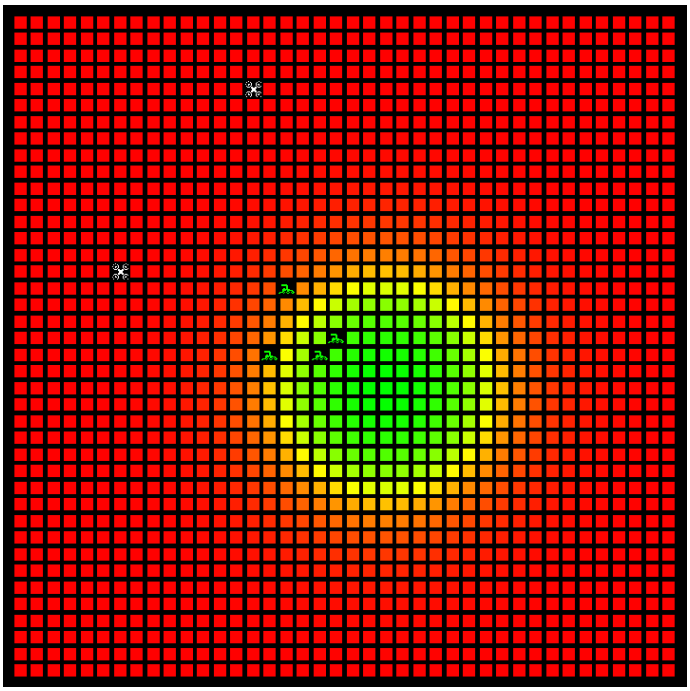
This section also offers detailed descriptions of each environment's features, explaining their functions and the appropriate use of environment variables. It also includes links to our library on the Python Package Index (PyPI) and the official GitHub repository. These resources enable users to seek clarifications and submit suggestions.

2.4. Data Collection

The collection of training data was conducted through simulated SAR missions in the DSSE environment, employing both DQN and PPO algorithms, and the baseline greedy search for comparison. The training process for the algorithms involves various configurations of the environment, each using a 40x40 navigation grid and a multi-agent scenario with four agents. A key variable in these configurations is the dispersion increment, which was set to 0.1. The dispersion is the spread of the probability zone within the environment; a higher dispersion indicates a larger search area as the highest probability zone expands throughout the ocean waters. The dispersion increment quantifies the rate at which the probability matrix grows.

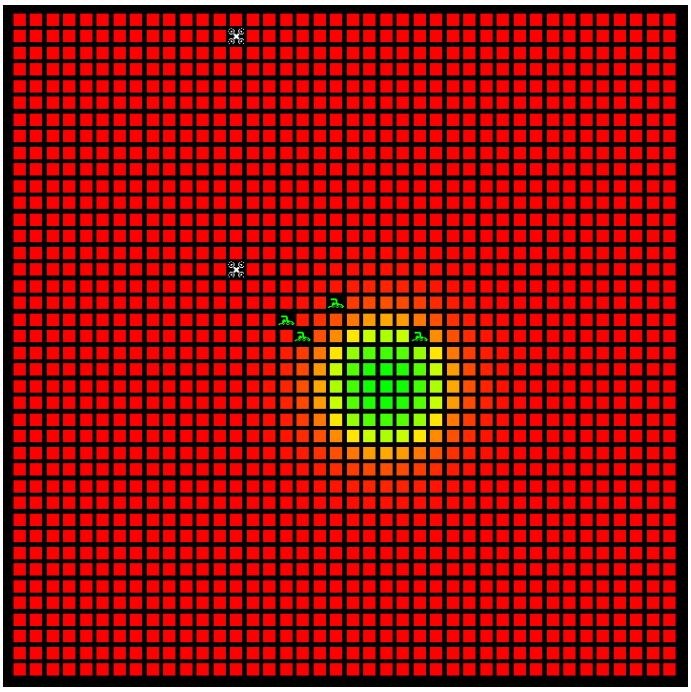
Figure 16 and Figure 17 demonstrate how changes in the dispersion increment affects the growth of the search area.

Figure 16: Simulation with 0.1 dispersion increment at 50 steps.



Source 16: Made by the authors.

Figure 17: Simulation with 0.05 dispersion increment at 50 steps.



Source 17: Made by the authors.

The exact tests made are described in section 2.6.

During the process, the training routine using RLlib [18] collected several training statistics. The two essential metrics that were used are: the number of actions executed by the agents and the total rewards received in each episode.

The collection of the validation data was accomplished by using a routine that executed the SAR mission simulation 5,000 times, employing the greedy algorithm along with the trained DQN and PPO algorithms across two neural network modes, within the same environment configurations described previously. The data collected include the rewards received, the number of actions realized and the information of whether the agents found the PIW on every trial by each algorithm.

2.5. Data Analysis

The analysis upon the collection of the training data was mainly accomplished by calculating the moving average of the rewards and actions, the size of the window for the moving average depends on the training batch, which is collected in parallel using RLlib and is then gathered in the Learner, who stores it to later analysis. The learning curve is a fundamental tool to evaluate the performance of a model, illustrating the rising of the received rewards while displaying a progressive reduction in the number of actions performed by the agents over the episodes.

2.6. Hypotheses

Five hypotheses were formulated to test the efficacy and learning rate of reinforcement learning algorithms, compared to a baseline informed algorithm, and comparing within the reinforcement learning algorithms themselves.

For all experiments conducted, we used the same base configurations: grid size of 40x40, 1 PIW, the dispersion matrix is centralized in the middle of the grid, 4 agents (drones) positioned around the dispersion matrix on 4 edges of the map, a dispersion increment of 0.1, and the default environment vector values, unless otherwise stated.

Hypothesis 1: Does reinforcement learning (RL) outperform a greedy strategy when the person in water moves away from the region of highest probability? This will be assessed by comparing the performance of agents controlled by a centralized algorithm (PPO and DQN) against a greedy policy under the specified base configuration.

We anticipate that RL will demonstrate superior adaptability and performance over the greedy approach, especially as the PIW's location deviates from predicted regions.

Hypothesis 2: Do agents trained with independent neural networks achieve faster convergence compared to those trained with shared networks? This will be investigated by analyzing the learning curve of agents in the base configuration, using a centralized and an independent learning approach.

The expected results for this hypothesis remain to be determined, as early experiment data suggests that the independent approach converges faster than centralized training. However, some literature works [24] indicates that centralized learning converges more quickly and enables better policy performance. Our goal is to ascertain which configuration is optimal for this problem and setup, thereby contributing to the research on this topic.

Hypothesis 3: What is the impact of trajectory information sharing among agents on search operations' effectiveness? This will be examined in an environment with large dispersion using the centralized version of the PPO algorithm. Two different approaches will be tested: one modifying observations with the agent's trajectory and directly altering probabilities on the matrix, and another utilizing a Long Short-Term Memory (LSTM) network to handle sequential information regarding agents' positions trajectories.

We predict that sharing historical search data among agents will enhance overall search efficiency, enabling a more coordinated and informed approach to the search strategy. Furthermore, recent studies have concluded that the usage of recurrent neural networks, such as the LSTM, in some partially observable environments [25] can boost the trained agent’s performance.

Hypothesis 4: Can agents be organized to ensure the full coverage is done in minimal time, while prioritizing the cells of highest probability, thereby maximizing coverage efficiency and minimizing search time? This will be explored using the coverage environment with 2 drones.

We hope to demonstrate that RL can effectively organize drone movements to avoid redundant searches, minimizing the SAR mission time and prioritizing regions that have a bigger chance of containing PIWs.

Hypothesis 5: Will the agents alter their search patterns when there are multiple Persons in Water (PIWs) compared to when there is only one PIW? This will be investigated by maintaining the base configuration while changing the number of PIWs to four.

It is expected that the drones will organize themselves to divide the task and search in different areas. This strategic distribution is anticipated to increase the coverage area, thereby enhancing the likelihood of detecting PIW and achieving a higher success rate.

3. Results

This section focuses on the results gathered by the team, of both the environment changes and recognition, and RL analysis, comparisons, and conclusions. The environment's enrichment allowed for a series of results including a paper application and recognition by the Farama team. Meanwhile, for the Reinforcement Learning results, there will be an analysis of the learning curves during the training phase, for each hypothesis, to demonstrate the progression of their decision-making capabilities, and later an analysis of the performance of the RL algorithms compared to baseline models where applicable, for each hypothesis given in the earlier paragraphs, with the intent to understand the behavior and the success rate of RL algorithms for each hypothesis given.

3.1. Environment Results

This sub-section will walk through the results of the developed training environments, which includes work done to the git repository to follow good practices of open source software, the publication of the environment on PettingZoo's [7] Third-Party environments, the submission of a paper about the environment on JOSS [8], as well as the client's approval of all changes made.

3.1.1. Open-Source Standards

To be able to submit a paper to JOSS, there are a list of requirements including necessary repository standards to align with good practices of open source software, withing this list, there is the inclusion of a code of conduct explicitly detailing good practices and behaviors for the community to follow, documentation that is available to all for free, a publicly disclosed guide on how to document and notify issues, and other factors. All the alterations and inclusions necessary to adhere to the policies stated by JOSS were made before the submission.

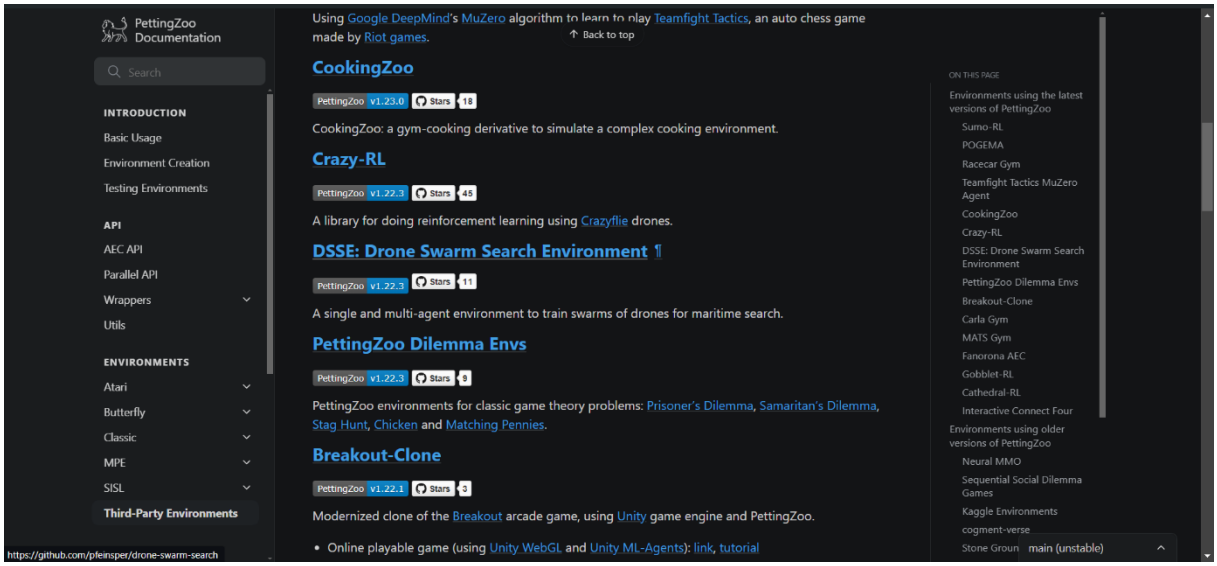
3.1.2. PettingZoo

PettingZoo [7] is one of many projects of Farama Foundation, a nonprofit organization with the mission of developing open-source reinforcement learning tools, created by the maintainers of OpenAI Gym as a replacement for the original OpenAI library [26]. Gym is an open-source project by OpenAI to develop reinforcement learning training environments and standards. "...It has been installed more than 43 million times via pip, cited more than 4,500 times on Google Scholar, and is used by more than 32,000 projects on GitHub. This makes it by far the most used RL library in the world" [26].

The Farama Foundation came into existence by the core maintainers of Gym, and currently it has been installed close to 69 million times [27]. Its biggest project, Gymnasium is the greatest standard of reinforcement learning environments. PettingZoo is defined by Farama as the gymnasium for multi-agent reinforcement learning environments.

As the developed environments are multi-agent and follow PettingZoo's defined interface, the team has been submitted and accepted to be part of the Third-Party environments list [28] (seen on Figure 18) on PettingZoo's repository, this give's greater visibility to the project, creating the possibility of other researchers to find and use the DSSE framework.

Figure 18: DSSE on PettingZoo's Third-Party Environments.



Source 18: PettingZoo's site [28].

3.1.3. Journal of Open Source Software

The Journal of Open Source Software [8] is an open-access free peer-reviewed journal dedicated to publishing articles describing OSS, the journal has a peer-reviewed process upon submission, with the intent to improve the quality of the code submitted. Since its establishment in 2016, JOSS has published close to three thousand peer-reviewed articles [29], with some works achieving hundreds of citations over the years [30].

Due to the scientific nature of the DSSE project, and its application in current state-of-the-art research, there was a submission made to the JOSS with a paper describing the DSSE library and it is two environments, the paper is currently under review.

3.2. Reinforcement Learning Results

The results for the Reinforcement Learning algorithms include the learning curves for each experiment done, as well as an analysis of their success rate and heuristic behaviors. Each experiment was conducted based on the previously stated hypothesis.

Before the final experiments, the team conducted a series of preliminary experiments using the Reinforce algorithm and an earlier version of the environment. These initial trials were designed to generate data for a paper submission to an IEEE conference. The experiments provided key insights to understand the agent's behavior within the environment and enhanced the team's understanding of RL algorithms. This foundational knowledge guided the team's approach for the subsequent stages of the project.

All experiments to test the five final hypotheses were done with both DQN and PPO algorithms. These tests adhered to the previously mentioned configurations: a map size of 40x40, four agents, a dispersion increment of 0.1, and a singular PIW in all tests except for those related to hypothesis 5, which involved four PIW's. The hyperparameters used for PPO for the experiments are described in Table 4, while the hyperparameters used for DQN are described in Table 5.

Table 4: PPO hyperparameters used for the experiments.

Parameter	Value	Description
B	8192	Training batch size
Lr	10^{-5}	Learning rate
γ	0.9999999	Discount factor
M	300	Stochastic Gradient Descent (SGD) minibatch
K	10	Number of SGD iterations

Table 5: DQN hyperparameters used for the experiments.

Parameter	Value	Description
B	512	Training batch size
Lr	10^{-4}	Learning rate
γ	0.9999999	Discount factor
U	500	Update target network every U steps
e_0	1	Initial epsilon for $\epsilon - greedy$
e_f	0.1	Final epsilon for $\epsilon - greedy$
T	400000	T timesteps for epsilon decay from e_0 to e_f.

3.2.1. Early Experiments

All initial results were obtained using an older version of the environment, which featured a simpler movement pattern for PIWs, a different reward scheme, and an alternate calculation for the probability matrix. During the initial training phase, the environment utilized a variable called dispersion constant, representing the maximum size of the probability matrix. This was later adjusted to the dispersion increment. All subsequent modifications aimed to enhance the realism of PIW movement.

For the experiments, there were four conducted, all with a grid size of 20 x 20, a singular agent and PIW. Two of the experiments used a dispersion constant of 1, representing a small probability matrix, and two with a dispersion constant of 5, representing a large probability matrix. The intent was to compare centralized and decentralized training, in both dispersion constants of 1 and 5. All experiments were compared to a simple greedy approach and were done with the objective of understanding if there were any differences in convergence time between training approaches and to compare the trained algorithms performance to a heuristic.

During these initial investigations, there was learning observed for both test groups, and a convergence in similar max rewards for both, approaching the theoretical max reward value for the environment at the time. There were differences in the convergence time, with the decentralized training converging in about one third of the steps taken, with seven hundred thousand steps for the decentralized algorithm in DC 1 and a million steps for the centralized

version in the same DC, and a million steps for the decentralized approach versus tree million steps for centralized version in a DC of 5; the team also observed that the centralized training was more unstable, with significantly higher peaks, and lower troughs throughout all the training stages.

To compare the RL algorithm to the greedy approach, a success rate test was devised. Each trained neural network underwent five thousand simulated missions; to calculate the mean and average success rate, the same number of tests were done for the greedy algorithm.

Table 6: Analysis of algorithms - Success Rates and Action Efficiency with Dispersion Constant 1.

Metric	Reinforce	Reinforce Independent Learning	Greedy Search
Success rate	86.99%	82.92%	100%
Mean Number of Actions	13.56	23.1	11.83
Median Number of Actions	12	13	11

Table 7: Analysis of algorithms - Success Rates and Action Efficiency with Dispersion Constant 5.

Metric	Reinforce	Reinforce Independent Learning	Greedy Search
Success rate	92.03%	92.71%	100%
Mean Number of Actions	18.46	23.56	16.89
Median Number of Actions	14	17	15

The tables above present the results of the experiments. As shown, the greedy approach achieved a 100% success rate in both scenarios. In contrast, the Reinforce algorithms attained success rates of eighty and ninety percent for scenarios with dispersion constants of 1 and 5, respectively. The success of the greedy algorithm is attributed to inaccuracies in the environment then. A simplistic movement pattern kept the PIW centered within the probability

matrix, which biased the environment in favor of the greedy approach. Consequently, the Reinforce algorithm was trained under an unrealistic scenario that did not accurately reflect the chaotic maritime environment.

These results provided the team with valuable insights into the current state of the environment, enabling them to focus their efforts on areas that would bring the most impactful outcomes in future work.

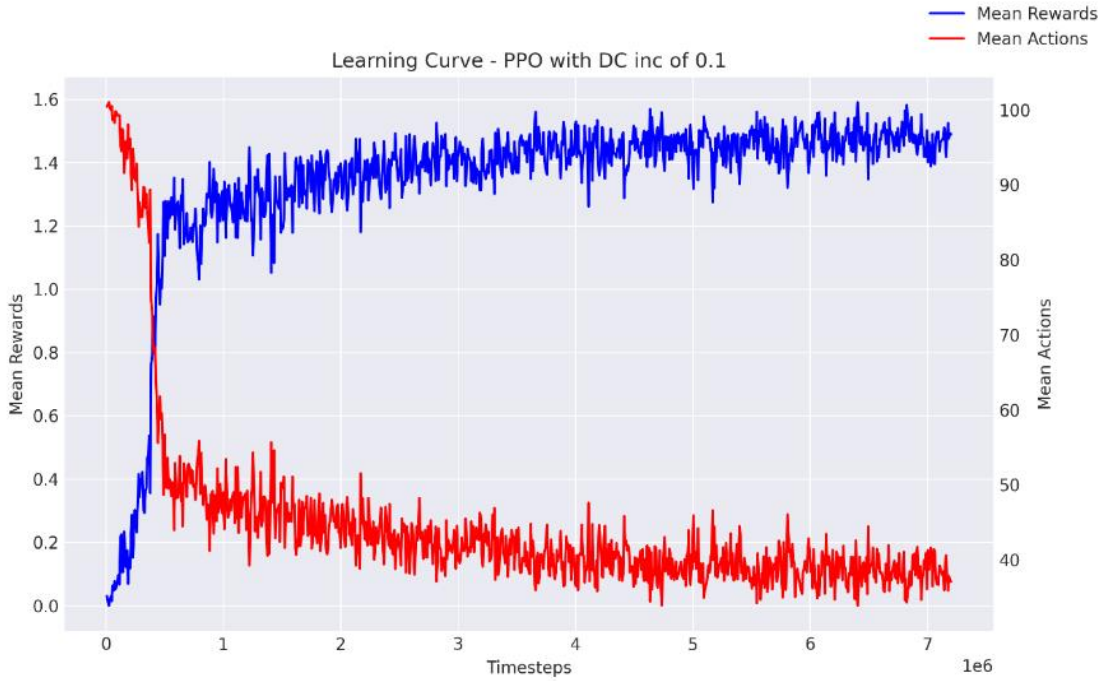
3.2.2. Final Results

With the early results, the modifications of the environment, and the new features on the algorithms, the team collected results regarding each hypothesis listed previously. The conclusion to each of these hypotheses and some considerations regarding them are going to be discussed in this section.

Hypothesis 1: Does the RL algorithm outperform the greedy strategy in scenarios where the PIW can move out of the region of highest probability?

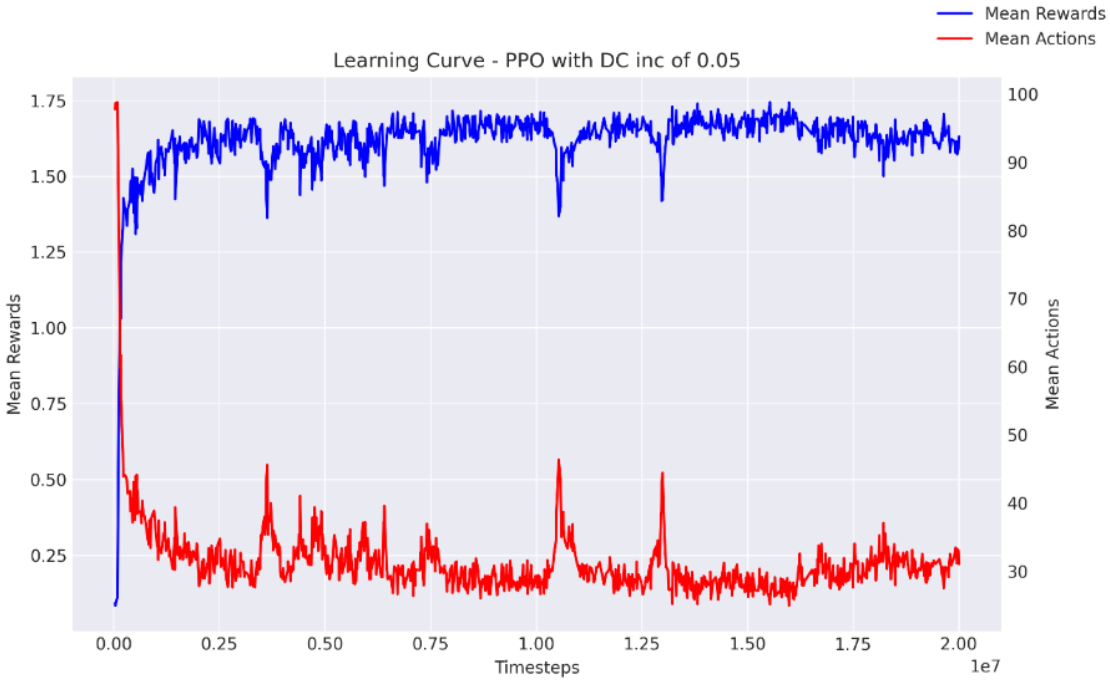
With the tests made using the RL algorithms on a 40x40 grid, varying only the dispersion increment of 0.1 and 0.05, we have concluded that the Reinforcement Learning algorithms can in fact outperform the greedy approach in environments that have a more complex movement for the targets. To support this statement, Figure 19 and Figure 20 present the learning curves for the training of PPO on the described environment configurations. Furthermore, in Figure 21 a comparison of PPO and DQN learning is provided.

Figure 19: Learning curve for PPO on 0.1 dispersion increment configuration.



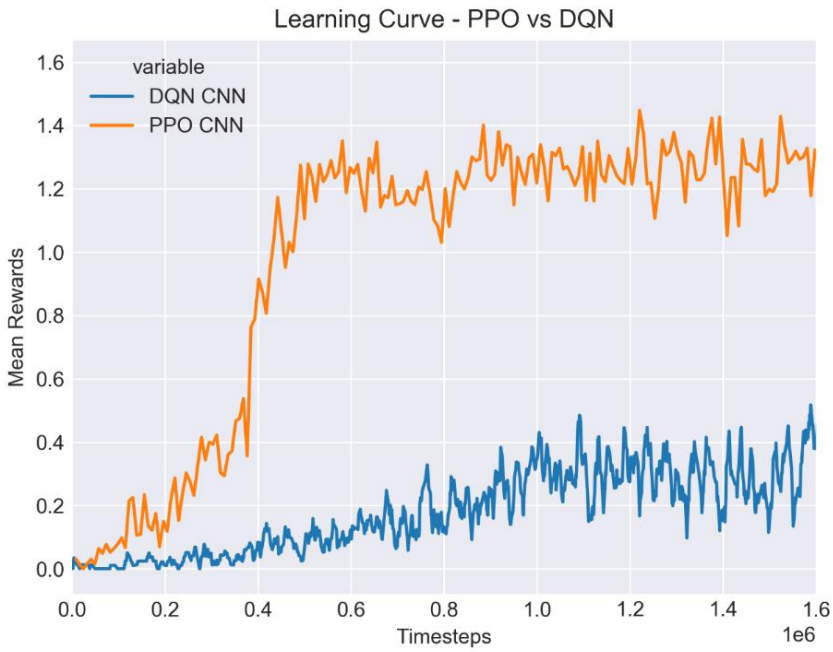
Source 19: Made by the authors.

Figure 20: Learning curve for PPO on 0.05 dispersion increment configuration.



Source 20: Made by the authors.

Figure 21: Learning curve comparison of PPO and DQN on 0.1 dispersion increment configuration.



Source 21: Made by the authors.

As can be seen from Figure 19 and Figure 20 where the mean reward is shown in blue, and the mean actions are shown in red, PPO manages to learn relatively quickly, reaching a reward in the range of 1.4 to 1.55 in the configuration with dispersion increment of 0.1 and a range of 1.5 to 1.75 on the configuration with dispersion increment of 0.05.

Another factor that must be observed, is that, as can be seen in Figure 21, on the same number of steps, the PPO algorithm learns a much better policy than DQN, having reached an average of 1.4, while DQN only reaches 0.4. It must be pointed out that for reaching this amount of timesteps, PPO training took 3h, while DQN took 23h. We hypothesize that the way DQN explores the environment, and its experience replay, makes it harder to learn a good policy due to the sparse nature of the environments rewards, as DQN uses $\varepsilon - greedy$ to randomly choose an action in random states, in contrast to PPO that samples the actions from its action probability distribution output, making it explore less on states that he was trained, and knows more, and exploring more on states he has less knowledge about. Given the difference between the two algorithms, the team decided to use PPO to test the other hypotheses.

In Table 8 and Table 9, is expressed the results of the algorithms tested 5000 in each configuration.

Table 8: Comparison of algorithms for 0.1 dispersion increment.

Metric	PPO CNN	Greedy
Mean reward	1.34	0.59
Action mean	42.47	77.48
Action median	23	100
Success (%)	75.44	35.84

Table 9: Comparison of algorithms for 0.05 dispersion increment.

Metric	PPO CNN	Greedy
Mean reward	1.48	0.86
Action mean	35.91	65.07
Action median	22	94
Success (%)	83.00	50.18

In Table 8 and Table 9, on the new version of the environment, where the movement of the PIW is more complex, and its position isn't always contained in the center of the probability matrix, the PPO Reinforcement Learning algorithm far outperform the greedy approach, having a difference of 75.44 % of success to 35.84 % in the configuration with the higher increasing dispersion, and a difference of 83 % to 50.18 % in the configuration with the lower increasing dispersion. We can attribute this to the idea that RL algorithms can learn more complex search patterns and are suitable for SAR missions that are less predictable and more complex, such as real-world SAR missions.

To get these results with the PPO algorithm, several tests were made, varying several factors, such as neural network architecture, training hyperparameters and even environment rewards. A factor that was key to getting better results was the alteration of the reward shape of the environment, that, as mentioned before, had rewards designed in a way that tried to teach the agents to first walk, then to search in the greatest probability cells, and then find the target. But, during our tests, we found that giving out sparse rewards for this problem achieves greater success, having the success rate of our agent stuck at approximately 50% before, and achieving 75.44 % after the modification. The motive for this is that the environment was not teaching the agents exactly the wanted behavior: finding the target, it was teaching several things, so the agent was stuck at an induced suboptimal behavior.

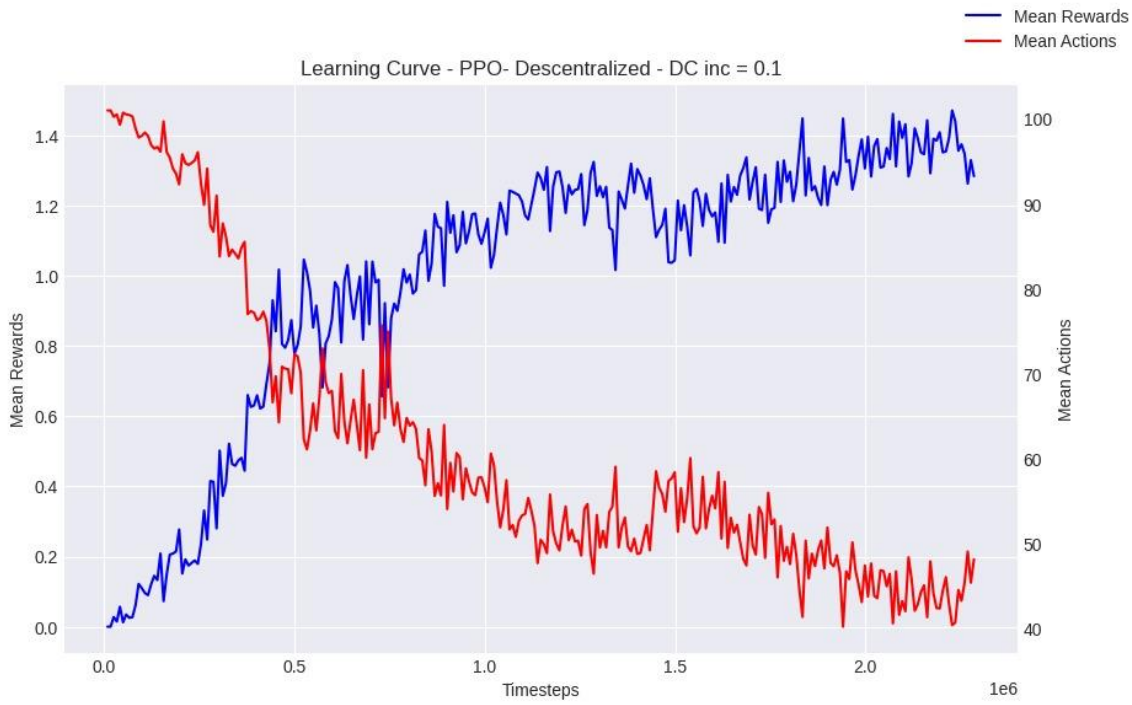
Hypothesis 2: Does decentralized training improve convergence time in a meaningful way and improve stability during training?

For the second hypothesis the team trained a PPO algorithm with a decentralized approach, meaning instantiating a policy per agent, instead of the conventional method of a single shared policy among all agents. This hypothesis came from literary review, as some peers stated in recent research that it does improve training time and stability, while others saw no difference or even a decrease in both time, stability, and trained policy performance.

The training done was identical to the ones done for H1 with PPO for a dispersion increment of 0.1, with the singular difference being the unique policies per agent, this allowed for a direct comparison to the standard PPO and the baseline algorithm.

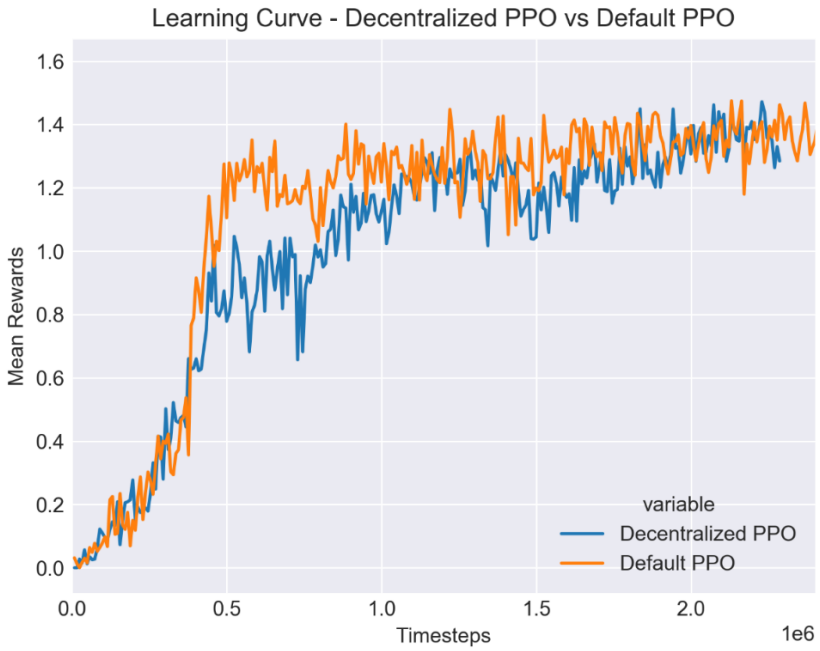
Since the preliminary results showed a positive tendency for lower training times and higher stability, the expectations for the results continued to be positive. With the results obtained, we concluded that decentralized training slows down the training time, decreases stability during training as well as converges to a worst policy, as can be seen by Figure 22 below, that shows the mean rewards and mean actions for the decentralized training, and Figure 23 showing the mean rewards for both the centralized and decentralized training for PPO.

Figure 22: Learning curve for decentralized PPO and dispersion increment of 0.1.



Source 22: Made by the authors.

Figure 23: Learning curve comparison Centralized PPO vs Decentralized PPO.



Source 23: Made by the authors.

In the figures above, there is a clear learning pattern for both version of the algorithm, with both achieving a mean reward around 1.4 by the end of training, with the centralized achieving this peak between five hundred thousand and a million steps while the decentralized version was able to arrive in the same mean reward around two million steps, and mean actions around 40 for the centralized version, and 50 for the decentralized version, indicating a clear decline and both metrics for the decentralized version.

Table 10 shows the test results for the PPO centralized and decentralized approaches, the validation tests were the same as the ones used for hypothesis 1.

There is a clear tendency of higher mean rewards, lower action means and median, and higher success rate for the centralized solution, in other words, the centralized solution proved unequivocally superior in all metrics, for this reason the team chose to continue forward with a centralized approach for all hypothesis testing going forward.

Table 10: Comparison of decentralized vs centralized PPO.

Metric	PPO Centralized	PPO Decentralized
Mean reward	1.34	0.94
Action mean	42.47	60.22
Action median	23	41
Success (%)	75.44	53.22

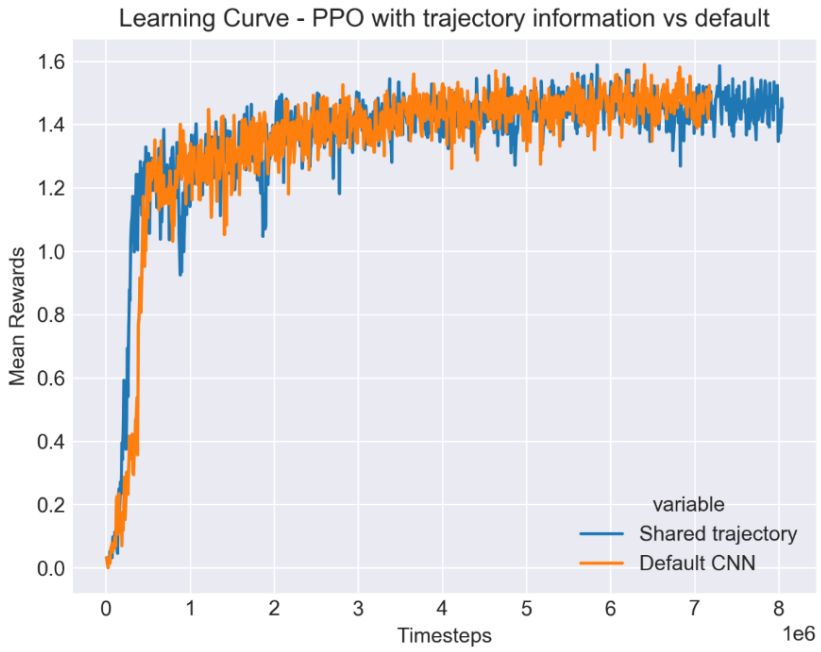
The experiments necessary to achieve these results were done using the team's implementation of RLlib library, with the same sparse reward shaping done for hypothesis one. The longer training steps can be explained by the extra resource use of multiple policies, as each policy instantiates a separate complete neural network architecture, meaning that a decentralized learning approach, with four agents uses four times the resources for the algorithms themselves compared to a centralized approach, this has impact on hardware and time resources, as training more neural networks takes more time and imposes the need to store more objects in memory. While the lower mean average is theorized be from the agents learning a sub-optimal policy trough it's training, as there could be a case of not all agents learning an ideal policy trough training, for a complex problem with a potentially small number of agents to solve, a singular agent learning a sub-optimal policy has potential to significantly impact the results.

Hypothesis 3: Does the information of the search trajectory impact on the search's results?

For the third hypothesis, a centralized PPO was trained with two different approaches, one modifying the observations with the agent's trajectory, where the probability of a cell on the probability matrix was directly modified based on the trajectory of the drones, decreasing it significantly right after a drone searched there, and linearly increasing with the timesteps after, until it becomes the original value. The other approach was done using a Long Short-Term Memory (LSTM) layer to handle the sequential nature of the information of the agents positions trajectory, this layer was inserted at the place of the second FC of the positions branch in the neural network Figure 10.

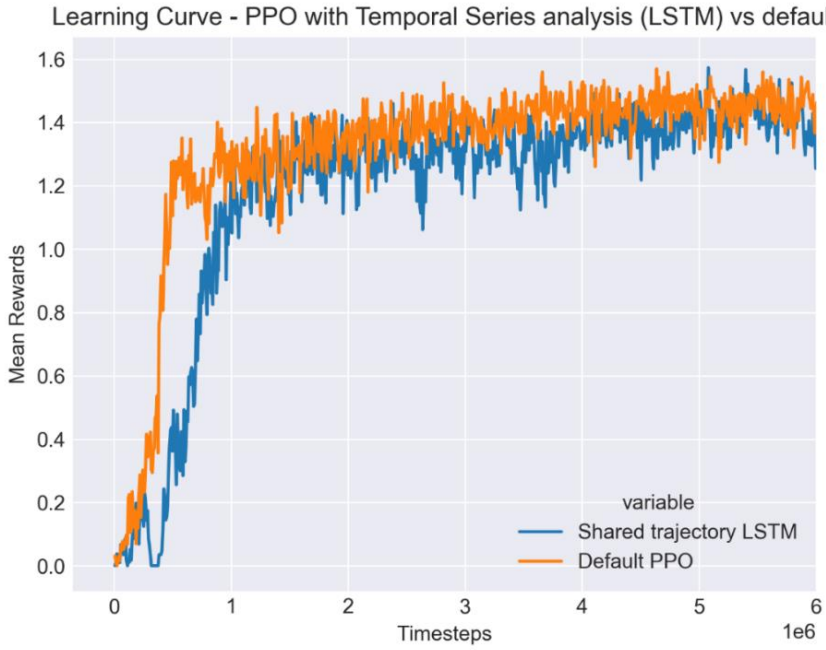
After conducting both tests, it was found that these modifications did not significantly impact search performance, as illustrated in Figure 24, Figure 25 and Table 11.

Figure 24: Learning curve comparison - PPO with trajectory information (matrix) vs default.



Source 24: Made by the authors.

Figure 25: Learning curve - PPO with LSTM vs default network.



Source 25: Made by the authors.

As it can be seen in Figure 24 and Figure 25, the learning with the trajectory being informed in the probability matrix, the learning using the LSTM layer and the learning with the default PPO configuration were all very similar. It is not possible to perceive any significant

differences to the learning progress. In Table 11, is possible to see the statistics of 5000 environment simulations for each of the approaches, and it makes it clear that the trajectory sharing methods did not cause a significative impact on the search performance.

Table 11: Comparison with trajectory sharing methods PPO and default PPO.

Metric	PPO default	PPO trajectory matrix	PPO LSTM
Mean reward	1.34	1.35	1.36
Action mean	42.47	41.99	41.57
Action median	23	23	23
Success (%)	75.44	75.98	76.46

The team hypothesizes that the information contained in the environment's observation is enough so that the algorithm's Markov Decision Process (MDP) [31] can choose a good action. So that the additional information ends up not adding nor subtracting from the agent's performance. During an interaction with the client, he suggested the hypothesis that the trajectory information could be more meaningful to the MDP if it was in a bigger scale search, mainly on a scenario with more agents, where the overlap of the search area would be bigger, thus having the information of previous searched areas could have a greater impact.

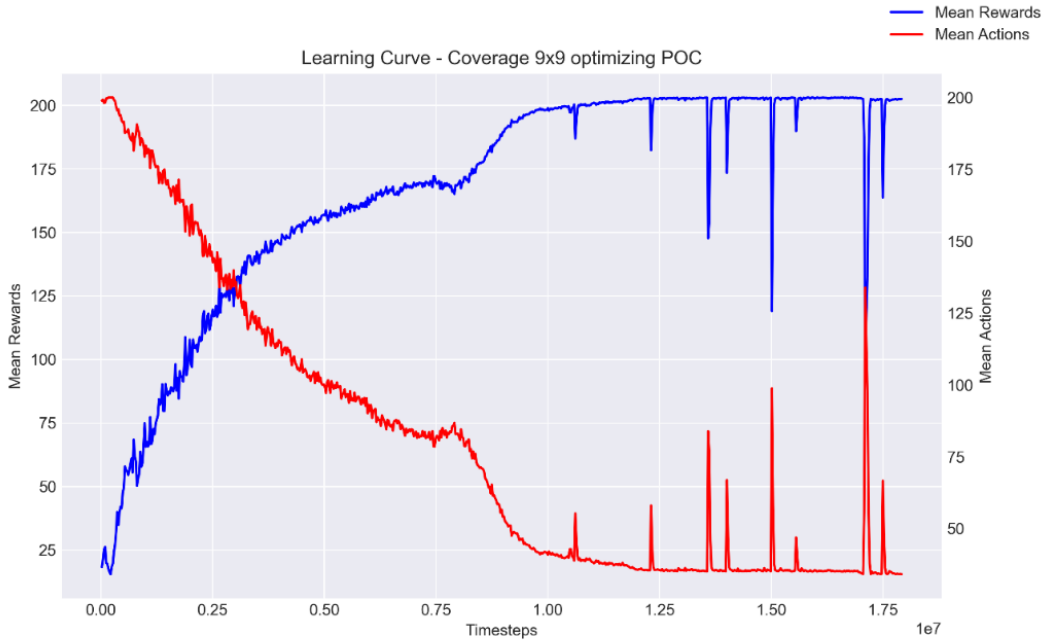
Hypothesis 4: Can the agents organize themselves to maximize coverage area and speed?

For this hypothesis, the coverage environment was used to replicate and extend the experiments done in the state of the art [9] [13], where the agents were put to learn to optimize a NP-Complete [21] problem and learn good solutions for covering the whole search area, minimizing the steps needed and prioritizing probability of containment. The experiment for this hypothesis was done in a 9x9 grid with 58 cells containing POC greater than 0, this matrix was generated with 2 hours of drifting simulation, using the default initial points of the library, that point to the ocean near Guarujá, SP. The agent's setup was done using two agents with a centralized multilayer perceptron that receives the whole probability matrix and the drone's positions, this approach drew better results than using the default CNN, most likely because of the small size of the used grid.

The trained agents learned to not only complete the search area, but prioritize the cells with greater POC and to make the search quicker (seen in Figure 26 with the decrease of mean actions), as the agents took an average of 34 steps to complete the 58 cells, meaning that they

learned to parallelize the search, doing it in only 58.62 % of the required steps, reaching almost the ideal 50% of speed up.

Figure 26: Learning curve for PPO optimizing coverage speed and POC.

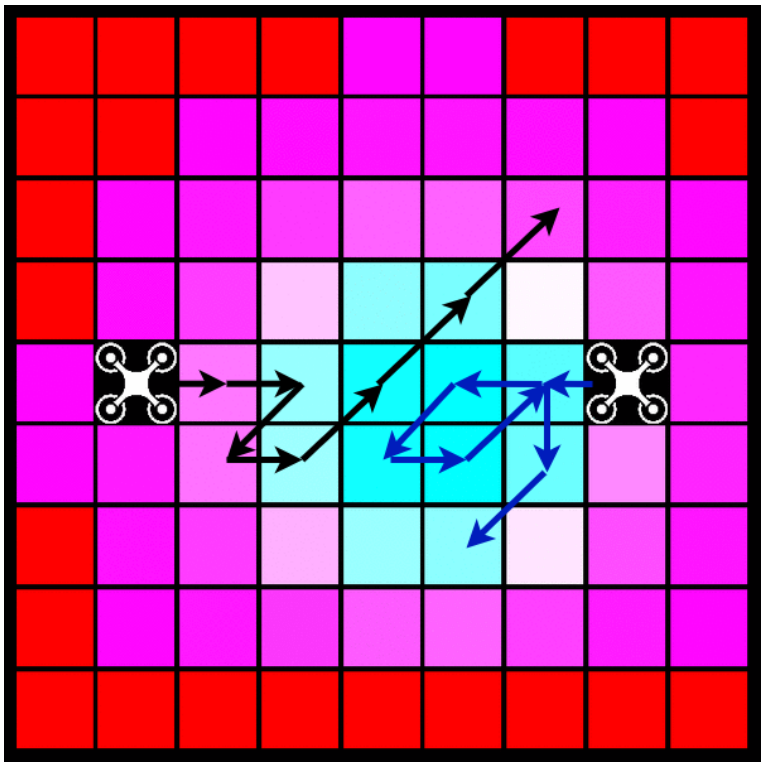


Source 26: Made by the authors.

As stated before, this experiment aimed to extend state-of-the-art research by increasing the complexity and using a multi-agent approach. The experiment utilized 58 valid search grid cells, compared to the 19 cells shown in [9] and 30 in [13]. The goal was to determine if agents could autonomously learn this complex task on a greater scale and organize themselves to parallelize the task effectively. The results indicate that the agents learned efficient behaviors and completed the task in less time than it would take a single agent. Additionally, this experiment highlights that reinforcement learning algorithms can handle multi-objective problems and learn tasks with more than one optimization, while prioritizing tasks that yield the higher sum of discounted rewards.

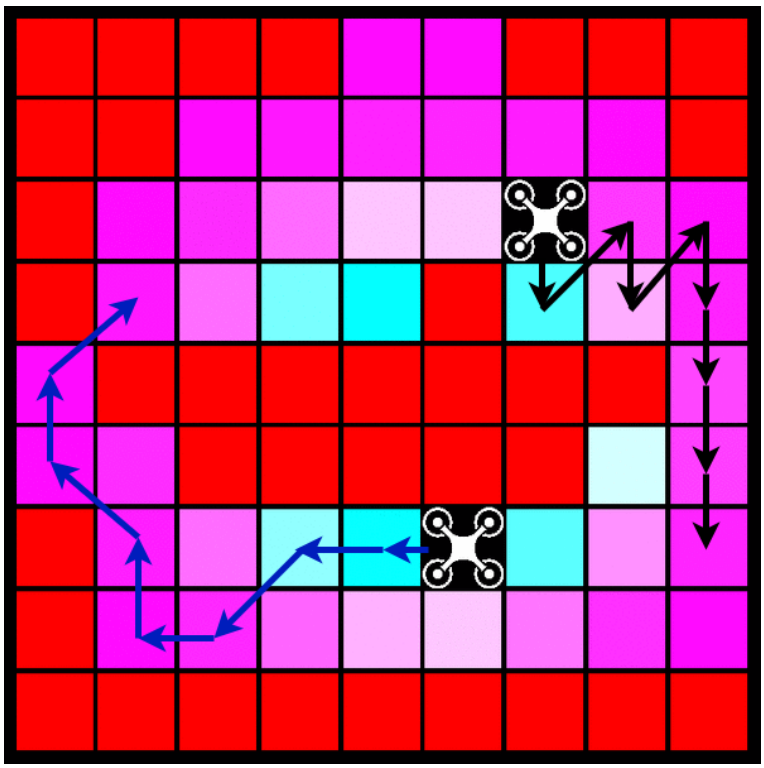
The figures below demonstrate the search pattern learned by the drones for the described scenario.

Figure 27: Image demonstrating the trained agent's behavior.



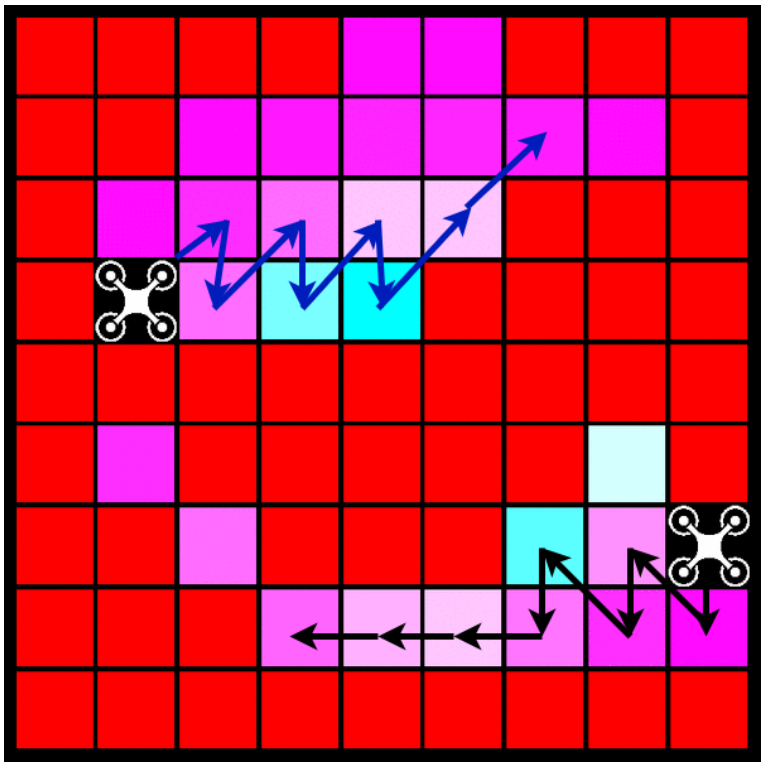
Source 27: Made by the authors.

Figure 28: Image demonstrating the trained agent's behavior.



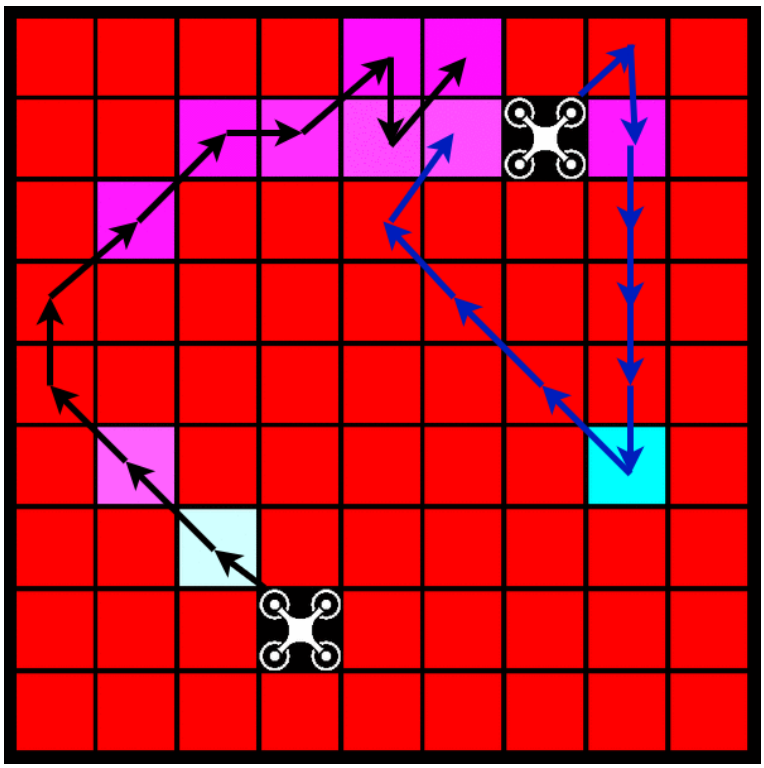
Source 28: Made by the authors.

Figure 29: Image demonstrating the trained agent's behavior.



Source 29: Made by the authors.

Figure 30: Image demonstrating the trained agent's behavior.

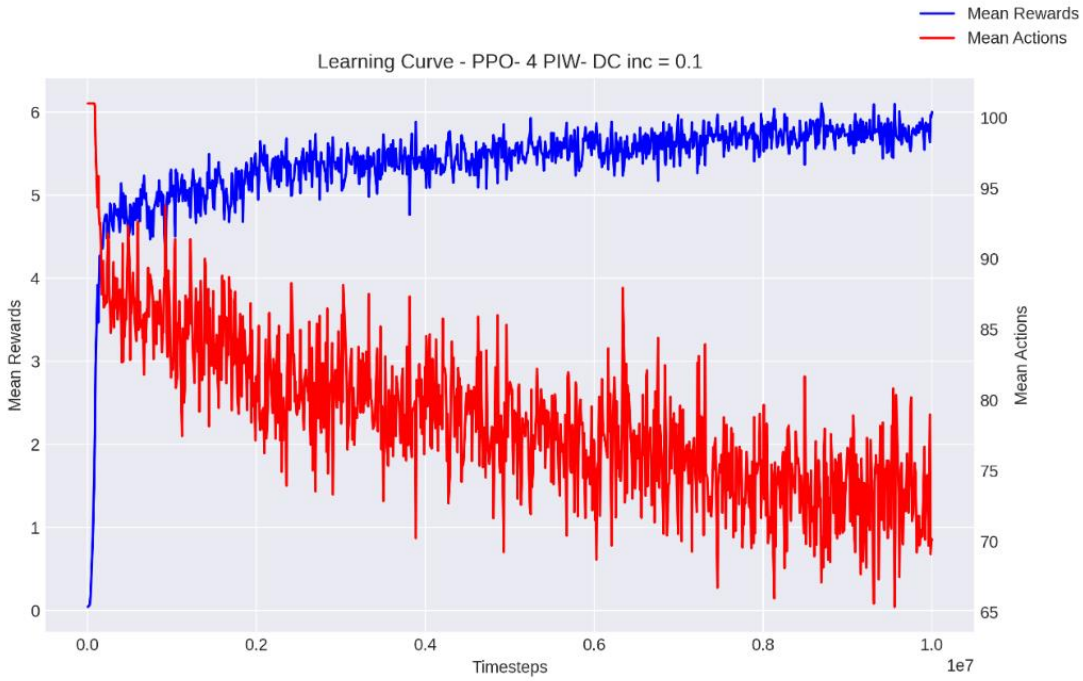


Source 30: Made by the authors.

Hypothesis 5: Will the agents alter their search patterns when there are multiple Persons in Water (PIWs) compared to when there is only one PIW?

To test the fifth hypothesis, experiments were conducted using the same control parameters as the other hypotheses, employing a PPO with a convolutional neural network and the sparse reward scheme. The only difference was the inclusion of four PIWs instead of one. The objective was to determine if the agents would adapt their behavior to account for the movement patterns of multiple PIWs.

Figure 31: Learning curve for PPO with 4 PIW.



Source 31: Made by the authors.

Figure 31 depicts the learning curve of the agents in a scenario with four PIWs. There is a clear learning trend, with mean actions decreasing as mean reward increases, indicating that the agents are effectively learning to coordinate their search for multiple PIWs.

In this scenario, the theoretical maximum reward is 8. However, the agents never reach a mean reward value of 8, instead converging around 6 after approximately ten million steps. By the end of the training, mean action oscillating significantly between 70 and 80. This instability in actions and rewards can be attributed to the increased complexity of managing the movements of four distinct PIWs, each with its own movement pattern within the probability matrix.

Table 12: Evaluating results for PPO with 4 PIW.

Metric	4 PIW
Found all PIW (%)	21.54
Average number of PIW found	2.3
Action mean	86.81

As shown in Table 12, the agents were able to find all PIWs in only 21.54% of the tests, a lower success rate compared to the achieved in other hypotheses. This outcome can be attributed to the learned movement pattern, where agents quickly move towards the center of the probability matrix and then follow it across the map. This behavior may arise from various factors, including the neural network architecture, the chosen RL algorithm, or the hyper-parameters used during training for this hypothesis.

Another possibility is that further research is required in this field, along with implementation of more advanced RL features. As the environment becomes more realistic, its stochastic nature demands specialized solutions to effectively manage the noisy and sparse conditions characteristic of the real world.

The agents found an average of 2.3 PIW per test, showing their ability to coordinate themselves to locate multiple PIWs. Although this rate is not sufficient in real-world scenarios where human lives are at stake and every PIW must be rescued, the team considers this result a success in terms of RL research. Nevertheless, further work is necessary to achieve a 100% success rate in all tests, ensuring all PIWs are consistently located.

3.3. Client feedback

The enhancements made to the environment have not only facilitated promising outcomes, including a paper submission and recognition from the Farama team, but have also garnered direct approval and agreement from the client regarding the results achieved. The client confirmed that the modifications led to a more realistic simulation environment compared to the initial version, effectively addressing their specific requests. Furthermore, the client expressed satisfaction and enthusiasm regarding the significant improvements observed in the learning capabilities of the algorithms. They particularly enjoyed the resulting coverage environment, noting that it was a great accomplishment to have incorporated a simulation that uses real-world data due to the potential future research it enables. This positive feedback underscores the success of the project in meeting the client’s expectations and advancing the field of Reinforcement Learning within the application domain.

4. Conclusions and Future Work

As with any research project, there are opportunities for further improvements and exploration in future iterations. This section will present the conclusions drawn from the work done and the hypotheses tested. Each hypothesis will be discussed individually, followed by a general conclusion. Additionally, the team will propose future work aimed at optimizing path planning for maritime search and rescue missions.

On the environment front, the DSSE framework was extended to two distinct environments with different mechanics. The search environment enhances the previous version of the project [2] by adding features such as more complex PIW movement, scaling the timestep with real-world time, incorporating more relevant SAR configurations, and revising the calculations of the probability matrix to enhance performance.

The coverage environment was developed to enhance state-of-the-art research regarding maritime coverage search path planning. This environment integrates a Lagrangian particle model [6] to simulate PIW drift in the ocean, making use of real maritime and wind data to construct the probability matrix, and extends existing work by introducing multi-agent capabilities.

The open-source environments [16] are believed to be valuable tools for researchers studying not only autonomous agents for SAR but for any application aimed at finding targets drifting in the ocean. They support research into using RL or other algorithms to control drones.

In the first hypotheses, the objective was to test if RL techniques do have superior adaptability and performance for the previously stated task, compared to a pre-determined, informed algorithm, with the focus on scenarios where the PIWs deviate from the regions of highest probability. After analyzing the results gathered and stated above, the team concludes that RL algorithms prove themselves superior in all metrics, for both scenarios of high and low dispersion increment.

In an extensive training conducted on a 40x40 grid, varying the dispersion increment between 0.1 and 0.05, the DQN algorithm did not learn an effective strategy. However, the PPO algorithms showed better results compared to the greedy approach, as evidenced by Table 8 and Table 9. The PPO algorithms outperformed the greedy algorithm with a success rate of 75.44% for a 0.1 increment, compared to 35.84% for the greedy approach. Similarly, for a 0.05 dispersion increment, PPO achieved a success rate of 83%, while the greedy algorithm attained

50.18%. Therefore, although the DQN algorithm failed to learn effectively, the PPO algorithm demonstrated superior performance over the greedy.

For the second hypothesis, the intent is to understand the effect of independent networks in convergence time and training stability. According to the results obtained, there is a clear trend of worse training time and increased instability during training of decentralized networks compared to centralized ones.

The need for extra training steps can be explained by the extra resource use of multiple policies, as each policy trains its own neural network, meaning that a decentralized learning approach with four agents uses four times the resources for the neural networks compared to a centralized approach, this impacts hardware resources as well as time resources, as training more neural networks takes more time. The lower mean average is theorized to be from the agents learning a sub-optimal policy, as there could be a case of not all agents learning an ideal policy, for a complex problem with a potentially small number of agents to solve, a singular agent learning a sub-optimal policy significantly impacts the results.

On the third hypothesis, the objective was to see the effects of historical search data among the agents in the search efficiency. The data gathered points in the direction of no change in behavior or search efficiency compared to no use of historical data for the agents.

The team concluded that the information contained in the environment's observation is enough so that the algorithm can choose a good action. So, the additional information ends up not impacting the agent's performance. The team theorizes that on missions where the overlap in the agent's search area is bigger, this information has a greater impact on the mission's success.

The fourth hypothesis focuses on efficient path planning for search operations over a given area, aiming to minimize redundant searches and search time while prioritizing areas with higher probability of containment. The results showed that the trained agents learned to complete the search area, prioritize cells with greater POC and accelerate the search process. On average, the agents took 34 steps to cover the 58 cells, demonstrating that they learned to parallelize the task and complete it in only 58.62% of the required steps, approaching the ideal 50% speedup. Additionally, this result indicates that reinforcement learning algorithms can

handle multi-objective problems and learn tasks with multiple optimizations, while prioritizing tasks that yield the higher sum of discounted rewards.

Finally, the fifth hypothesis aims to understand the organization and movement patterns agents adopt in scenarios with multiple PIWs. The results showed that the agents organize themselves to locate multiple PIWs in SAR scenarios. Although they are not yet able to find all PIWs in most tests, they consistently locate more than half, which the team considers a success in research terms.

The results indicate that, although the outcomes were not optimal, the algorithms are indeed capable of learning the intended search patterns, with the agents positioned in the middle of each side of the grid, the agents rush toward the probability matrix and anticipate is movement in order to intercept it at the right place, then the agents start search around the center and behind of the probability matrix until they find the PIW.

The learning curve shows a clear trend of agents improving their coordination in searching for multiple PIWs, as evidenced by the increasing mean reward and decreasing mean actions. However, the convergence of agents on rewards lower than the maximum suggests that while the algorithms can effectively learn to manage multiple PIWs, the increased complexity of handling the distinct movements of multiple PIWs leads to difficulties in generalizing their behavior. Consequently, the algorithms arrive at a suboptimal solution when faced with larger quantities of PIWs.

Regarding the features and improvements to the environment as well as the research goals set up by the team, all of them were achieved during the research done, and a series of extra goals were pursued as well, this includes the submission of the paper to the JOSS and the submission of the paper for the IEEE Fusion conference.

Below is the full list of goals and features achieved and implemented throughout this report's work.

- Writing of an informed baseline algorithm to have a fairer comparison to the RL techniques.
- Training of RL algorithms with different environment configurations to observe and analyze how the agents behave compared to the baseline algorithm, including a single and multiple PIW, different grid sizes and different initial positions for the probability matrix.

- Experiments utilizing a single neural network for the entire drone swarm, and a single neural network per agent.
- Usage of new RL methods: Deep Q-Networks (DQN) [4] and Proximal Policy Optimization (PPO) [5].
- Adding intercardinal movement to agents to match current maritime SAR simulations.
- Establishing mechanisms for the exchange of information regarding areas already surveyed by the drones.
- Inclusion of the possibility of simulation for multiple PIW.
- Incorporating different movement speeds for the agents (drones) and targets, recreating real world differences between the two.
- A revision of the person's movement to better align with real-world research and more accurately reflect realistic conditions.
- Revisions of the probability matrix to better align with real-world research and more accurately reflect realistic conditions.
- Individual Probability of detection (POD) to better replicate the uncertainties of real-life SAR operations. The POD introduces a probability of not detecting the person, influenced by the weather conditions, drone altitude, and other statistics.
- Anchoring of the simulation time-step with real time, to facilitate research going forward.
- Inclusion of a pre-render time to replicate real world cases where rescue teams require time before arriving at the disaster area.
- Usage of a Lagrangian particle model [6] to replicate the complex movements and drift patterns of the ocean and model the trajectories of PIW more accurately.
- The creation of a second environment expanding on state-of-the-art research on RL maritime full coverage path planning.
- Rework of the reward function for the DSSE.
- Restructuring of the environmental architecture to fit withing PettingZoo’s [7] standards.
- Remove collisions between drones and changes so they cannot leave the map.
- Research done with the second environment to expand on current state of the art works.
- Deployment of documentation for the environment replicating the best open-source practices.

- Inclusion of automated testing and deployment through GitHub actions, replicating the best open-source practices.
- Formulation of hypotheses to train and test RL algorithms.
- Testing done to answer all hypotheses formulated.
- Deployment of documentation website, aligning it with good open-source practices.

For future works, more extensive research involving the 4th hypothesis is needed, as it is more closely related to real world conditions. This includes testing various RL strategies designed for environments with sparse rewards, such as curiosity-driven exploration [32], this is a concept where agents receive intrinsic rewards for exploring the environment. This form of RL exploration is considered ideal for stochastic, sparse environments where the agents need to learn specific policies in a world with distracting noisy variables. The approach of curiosity implementation might be a way forward to achieve better success rates for the agents.

Additionally, testing new RL algorithms in both environments and experimenting with scenarios where drones communicate with each other is essential to continue research to understand the impact of communication between the agents. If communication improves SAR missions' success rate, adding intermittent communication failures is necessary to further approximate to real-world conditions, where packet losses are common. Another idea is to analyze environments where the probability of detection (POD) is less than 1, meaning there is a chance that the drone will not locate the PIW in each cell, replicating real world conditions where the onboard sensors might not detect a PIW due to weather conditions, or sensor failure.

References

[1] D. Silver, S. Singh, D. Precup and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, vol. 299, October 2021.

[2] L. Abreu, L. Carrete, M. Castanares, E. Damiani, J. F. Brancalion and F. J. Barth, "Exploration and Rescue of Shipwreck Survivors using Reinforcement Learning-Empowered Drone Swarms," *XXV Simpósio de Aplicações Operacionais em Áreas de Defesa*, pp. 64-69, 2023.

[3] M. Castanares, L. F. S. Carrete, E. F. Damiani, L. D. M. de Abreu, J. F. B. Brancalion and F. J. Barth, "DSSE: A Drone Swarm Search Environment," 12 July 2023. [Online]. Available: https://arxiv.org/abs/2307.06240. [Accessed 17 March 2024].

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, pp. 529-533, 25 February 2015.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," 28 August 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1707.06347. [Accessed 12 April 2024].

[6] K. -F. Dagestad, J. Röhrs, Ø. Breivik and B. Ådlandsvik, "OpenDrift v1.0: a generic framework for trajectory modelling," *Geoscientific Model Development*, vol. 11, pp. 1405-1420, 13 April 2018.

[7] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. Santos, R. Perez, C. Horsch, C. Dieffendahl, N. Williams and Y. Lokesh, "PettingZoo: Gym for multi-agent reinforcement learning," 2021. [Online]. Available: https://github.com/Farama-Foundation/PettingZoo. [Accessed 9 May 2024].

[8] "Journal of Open Source Software," NumFOCUS, [Online]. Available: https://github.com/openjournals/joss. [Accessed 9 May 2024].

[9] J. Wu, L. Cheng, S. Chu and Y. Song, "An Autonomous Coverage Path Planning Algorithm for Maritime Search and Rescue of Persons-In-Water Based on Deep Reinforcement Learning," *Ocean Engineering*, vol. 291, p. 116403–116403, 1 January 2024.

[10] E. T. Alotaibi, S. S. Alqefari and A. Koubaa, "LSAR: Multi-UAV Collaboration for Search and Rescue Missions," *IEEE Access*, vol. 7, pp. 55817-55832, 22 April 2019.

[11] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229-256, May 1992.

[12] D. W. Schuldt and J. Kuruca, "Maritime search and rescue via multiple coordinated UAS," 2016.

[13] B. Ai, M. Jia, H. Xu, J. Xu, Z. Wen, B. Li and D. Zhang, "Coverage path planning for maritime search and rescue using reinforcement learning," 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801821014220. [Accessed 11 April 2024].

[14] G. E. Hinton, S. Osindero and Y. -W. Teh, "A Fast-Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, p. 1527–1554, July 2006.

[15] R. Camargo, "Metodologia Agile: quais as mais utilizadas?," RC ROBSON CAMARGO - PROJETOS E NEGÓCIOS, 2018. [Online]. Available:

https://robsoncamargo.com.br/blog/Metodologia-Agile-quais-as-mais-utilizadas. [Accessed 17 March 2024].

[16] L. F. Carrete, M. Castanares, E. Damiani, L. Malta, J. Oliveira, R. R. Rodrigues, R. L. Falcao, P. Andrade and F. J. Barth, "DSSE: An Environment to Train Drones to Search and Find a Shipwrecked Person Lost in the Ocean Using Reinforcement Learning," [Online]. Available: https://pypi.org/project/DSSE/. [Accessed 17 March 2024].

[17] R. C. Martin, Clean Code: A Handbook of Agile Software Craftmanship, Pearson Education, 2008.

[18] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," in *International Conference on Machine Learning*, 2018.

[19] "The Python Profilers," Python Software Foundation, [Online]. Available: https://docs.python.org/3/library/profile.html#module-cProfile. [Accessed 17 March 2024].

[20] "Numba: A High-Performance Python Compiler," 2018. [Online]. Available: http://numba.pydata.org/. [Accessed 17 March 2024].

[21] K. E. Trummel and J. R. Weisinger, "The Complexity of the Optimal Searcher Path Problem," *Operations Research*, vol. 34, p. 324–27, 1986.

[22] J. Torres, *Policy-Gradient Methods*, Towards Data Science, 2020.

[23] Ray, "RLlib docs," Ray, 2024. [Online]. Available: https://docs.ray.io/en/latest/rllib/index.html. [Accessed 12 05 2024].

[24] A. M. Labanca, *Deep Reinforcement Learning for Kamikaze Drone Decision-Making*, São José dos Campos: Instituto Tecnológico de Aeronáutica, 2024.

[25] X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng and J. He, *Recurrent Reinforcement Learning: A Hybrid Approach*, arXiv, 2015.

[26] The Farama Foundation, "Announcing The Farama Foundation," 25 October 2022. [Online]. Available: https://farama.org/Announcing-The-Farama-Foundation. [Accessed 15 May 2024].

[27] The Farama Foundation, "Installations," [Online]. Available: https://farama.org/stats/installations. [Accessed 14 May 2024].

[28] The Farama Foundation, "PettingZoo's Third-Party Environments," 04 05 2024. [Online]. Available: https://pettingzoo.farama.org/main/environments/third_party_envs/. [Accessed 12 05 2024].

[29] Journal of Open Source Software, "Journal of Open Source Software papers published," [Online]. Available: https://joss.theoj.org/papers/published. [Accessed 10 05 2024].

[30] A. M. Smith, K. E. Niemeyer, D. S. Katz, L. A. Barba, G. Githinji, M. Gymrek, K. D. Huff, C. R. Madan, A. C. Mayes, K. M. Moerman, P. Prins, K. Ram, A. Rokem, T. K. Teal, R. V. Guimera and J. T. Vanderplas, "Journal of Open Source Software (JOSS): design and first-year review," *PeerJ Comput Science*, 2018.

[31] S. V. Albrecht, F. Christianos and L. Schäfer, "Markov Decision Processes," in *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*, Cambridge, MIT Press, 2024, p. 22.

[32] D. Pathak, P. Agrawal, A. A. Efros and T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," in *International Conference on Machine*, Sydney, 2017.