Reinforcement Learning Applied to Train Autonomous Maritime Search and Rescue Drones

Jorás C. C. de Oliveira[1], **Pedro H. B. A. Andrade**[1], **Renato L. Falcão**[1], **Ricardo R. Rodrigues**[1], **José Fernando Brancalion**[2], **Fabrício J. Barth**[1]

[1]Insper. São Paulo,SP - Brazil

[2]Embraer. São José dos Campos,SP - Brazil

{jorascco,pedroa3,renatolf1,ricardorr7}@al.insper.edu.br

jose.brancalion@embraer.com.br, fabriciojb@insper.edu.br

Abstract. *This paper presents a Search and Rescue (SAR) environment tailored for locating shipwrecked individuals and evaluation of Reinforcement Learning (RL) algorithms under different scenarios, considering a variety of different hypotheses, and an extensive number of experiments. Our findings indicate that RL techniques, particularly Proximal Policy Optimization (PPO), significantly outperform traditional greedy algorithms regarding success rates. Centralized network architectures demonstrate superior convergence compared to decentralized ones. Historical search data does not notably enhance algorithm performance, suggesting that real-time observations are sufficient. Agents are able to naturally parallelize the search efforts within a given probability zone while prioritizing higher probability areas first. Finally, while managing multiple persons-in-water (PIWs) increases complexity, agents show effective coordination and improvement over time, underscoring the potential of RL in complex SAR missions. This study highlights the promising role of RL in optimizing SAR operations.*

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. According to several authors, using 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 are still in their infancy, it is theorized that introducing AI for SAR applications may significantly improve the efficacy of search operations and reduce the time needed to find and save PIW [Wu et al. 2024, Ai et al. 2021]. Reinforcement Learning (RL) is believed to enable the development of new, more efficient search patterns tailored to specific applications. This is based on the hypothesis that reward maximization is sufficient to foster generalization abilities, thereby creating powerful agents [Silver et al. 2021]. Such advancements could potentially lead to the saving of more lives.

To enhance the effectiveness of Unmanned Aerial Vehicles (UAVs) in Search and Rescue (SAR) missions, Alotaibi [Alotaibi et al. 2019] introduces the Layered SAR (LSAR) algorithm. This method starts by focusing on a central disaster location where

most survivors are likely to be, then gradually expands the search area. Utilizing cloud robotics, UAVs communicate with a central cloud manager, which orchestrates the operation. The disaster area is divided into square layers, each maintaining a list of survivors with their coordinates, updated on the cloud server. UAVs are assigned to these layers, sometimes with multiple UAVs per layer for enhanced efficiency.

In "Maritime Search and Rescue via Multiple Coordinated UAS" [Schuldt and Kurucar 2016], a coordinated strategy for UAVs in maritime SAR missions is presented. This involves partitioning the search area based on probability and assigning drones to specific sections, maintaining minimum distances between them. A greedy algorithm matches the most capable UAVs to significant partitions, factoring in endurance, camera quality, and survivor likelihood. The study also introduces an equation for calculating the visible area of each drone.

Ai et al. 2021 explores reinforcement learning (RL) in maritime SAR, using boats for the search. The study focuses on decision-making, search area determination, and maritime vehicle deployment, using metrics like the probability of detection (POD), probability of containment (POC), and probability of success (POS). A drift prediction component forecasts the trajectory of distressed objects, constructing a search grid with cells assigned a POC. Using Q-learning, the algorithm selects actions to maximize POS based on the current state. Experimental results show effective coverage with minimized redundancy.

Wu et al. 2024 builds on this with deep reinforcement learning (Deep RL), using the DQN algorithm, where deep neural networks replace the Q-table, allowing the algorithm to approximate values and choose the best action based on the environment. Experiments show this approach matches or exceeds the Q-learning results, completing SAR missions more efficiently. However, both methods assume a constant search area, not accounting for dynamic factors like wind or waves during the search.

The LSAR [Alotaibi et al. 2019], Q-learning [Ai et al. 2021], UAS [Schuldt and Kurucar 2016], and DQN [Wu et al. 2024] approaches all use probability regions to locate targets. LSAR consistently centers the highest probability region, while other methods use maritime data and statistical methods. LSAR and UAS employ multiple agents, with LSAR drones communicating upon detection and UAS agents operating independently. Each approach aims to optimize SAR missions and enhance success rates using distinct strategies.

Abreu et al. 2023 implemented a Reinforce algorithm that considers a dynamic map of probabilities, representing the chances of a person being found, as well as the position of other agents. The findings of this work provided a proof of concept, supporting the notion that reinforcement learning strategies outperform predefined paths in terms of efficiency and effectiveness. However, this work did not explore other aspects, like, state representation, neural network architecture, and data sharing between agents.

This paper aims to define a Search and Rescue (SAR) environment and evaluate and compare various reinforcement learning (RL) algorithms within these settings. Our objective is to experiment with and enhance real-world SAR operations using multi-agent reinforcement learning techniques, [Ai et al. 2021, Wu et al. 2024, Abreu et al. 2023] appear already to highlight the promising role of RL in SAR operations, and we will iterate

on said works demonstrating the potential for autonomous search parallelization as well as high recovery rates in tested SAR scenarios. This paper is structured as follows: in the section 2, we provide a detailed overview of a SAR environment; in the section 3, we review and compare different RL algorithms that can be applied to this environment; in section 4, we present our experimental setup and results; finally, we discuss our findings.

2. Environment

For this project, the Drone Swarm Search Environment (DSSE) framework, developed by the authors and available as a Python package [Falcão et al. 2024], was used to study the viability of using multi-agent reinforcement learning in searching for shipwrecked people. This environment is designed to train RL agents to locate PIW and 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.

The environment is a 2D grid with a probability matrix representing the probability of containing a PIW. The probability matrix is based on a Gaussian distribution and a dispersion increment, which controls the Gaussian expansion rate. The distribution is calculated using the bi-dimensional Gaussian function presented in the equation 1.

$$f(x, y) = A \cdot \exp\left(-\left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0)^2}{2\sigma_y^2}\right)\right) \tag{1}$$

where A is the amplitude of the Gaussian function, x_0 e y_0 are the coordinates of the supposed position of the PWI. σ_x e σ_y define how the function will be stretched along the matrix, and correspond to the dispersion increment, determining how quickly the Gaussian expands in both directions. Finally x e y represent the horizontal and vertical positions within the grid.

The dispersion increment dictates the rate at which the Gaussian expands. For instance, a value of 0.1 allow faster expansion, suitable for cases where the PIW moves rapidly, while 0.05 provides a slower, more precise spread. These values were chosen to balance coverage and accuracy in different scenarios

The direction of movement is controlled by a vector representing ocean currents and the wind. The PIW's movement is influenced by the probabilities in the matrix, selecting a pseudo-random direction weighted by the probabilities of adjacent cells and an independent movement vector.

The environment states are represented as a tuple of two boxes: one for the drone's position (x, y) and another for the probability matrix. This allows the agents to use the probabilities and their own positions to decide the best action, facilitating the orchestration of complex search patterns to find the PIW. The action space of the environment is discrete, consisting of 9 actions: moving in the cardinal and inter cardinal directions and searching the current cell. With a Probability of Detection (POD) of 1.0, the environment is deterministic, but with lower values, it becomes stochastic. The problem presents itself as a sparse reward problem, as the goal is reached only when the PIW is found, the reward scheme, represents this, as it can be seen in the equation 2.

$$R(Ts) = \begin{cases} 2 - \frac{Ts}{Ts_{limit}}, & \text{if found} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where Ts is the time step that the PIW was found and Ts_{limit} is the configurable time step limit.

3. Reinforcement Learning Algorithms

To effectively compare and evaluate multiple search algorithms, we implemented a baseline algorithm and utilized the RLlib [Liang et al. 2018] library to implement the reinforcement learning algorithms Proximal Policy Optimization (PPO) [Schulman et al. 2017] and Deep Q-Network (DQN) [Mnih et al. 2015], supporting both central and independent learning configurations [Albrecht et al. 2024]. Central learning applies single-agent RL to joint actions for a central policy, while independent learning applies single-agent RL to each agent independently, ignoring the presence of other agents.

DQN and PPO were chosen to represent the main algorithms from the value-based and policy-based families, respectively. The two algorithms differ in their learning and exploration strategies, which are crucial for the reinforcement learning process. PPO selects actions by sampling from the probability distribution output by its policy network, facilitating higher exploration when the agent is uncertain and greater exploitation as the agent learns. In contrast, DQN uses an ϵ-greedy algorithm, balancing exploration by sampling random actions and exploitation by choosing actions with the highest Q-value.

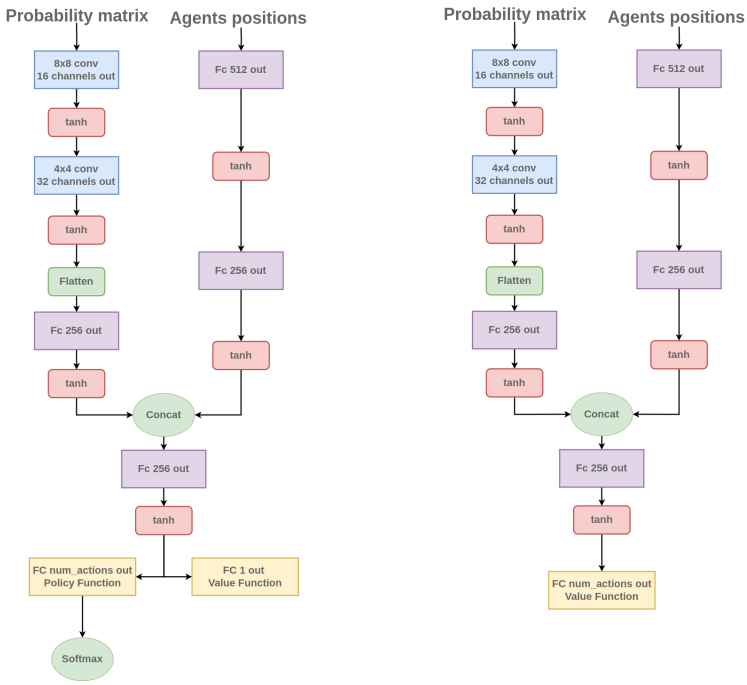
The baseline algorithm does not use reinforcement learning but leverages the probability matrix to guide agent actions. It follows a simple heuristic where each agent is assigned to search the cell with the highest probability, focusing solely on exploitation. This non-stochastic, easy-to-implement approach serves as a comparative tool to evaluate the efficacy of the RL algorithms.

3.1. Data Collection

The training data was collected through simulated SAR missions in the DSSE environment, employing 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 in a multi-agent scenario with four agents. A key variable in these configurations is the dispersion increment, set to 0.1 and 0.05. Dispersion is the rate at which the probability matrix spreads within the environment. A higher dispersion rate means the search area expands quickly.

The validation data was collected by running the SAR mission simulation 5,000 times. This involved the greedy algorithm and the trained DQN and PPO algorithms across two neural network modes in the same environment. The figures 1a and 1b describe the neural network (NN) used in the PPO and DQN implementation, respectively.

The training routine using RLlib [Liang et al. 2018] collected rewards received, the number of actions taken, and whether the agents found the PIW in each trial. The training data analysis involved calculating the moving average of the rewards and actions, with the window size depending on the training batch. Data was collected in parallel



(a) NN for PPO Implementation

(b) NN Diagram for DQN Implementation

Figure 1. Neural network architectures

using RLlib and stored by the Learner, the class within RLlib responsible for defining neural network training parameters and data gathering for later analysis.

3.2. Hypotheses

Four hypotheses were formulated to test the efficacy and learning capacity of reinforcement learning algorithms. For all experiments conducted, the same base configurations were used: grid size of 40x40, 1 PIW, the probability 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 or 0.05, and the default environment vector values unless otherwise stated.

Hypothesis 1 (H1): *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. It is anticipated that RL will demonstrate superior adaptability and performance over the greedy approach, especially as the PIW's location deviates from predicted regions.

Hypothesis 2 (H2): *Do agents trained with independent neural networks achieve faster convergence than 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 [Albrecht et al. 2024]. However, some literature indicates

that centralized learning converges more quickly and enables better policy performance [Labanca 2024]. We aim to ascertain which configuration and setup are optimal for this problem.

Hypothesis 3 (H3): *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 approaches will be tested: one modifying observation 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 using recurrent neural networks, such as the LSTM, in some partially observable environments [Li et al. 2015] can boost the trained agent's performance.

Hypothesis 4 (H4): *Do agents develop new search patterns when there is more than one PIW?* This will be investigated by changing the number of PIWs to four on the base configuration. 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.

4. Results

The results for the Reinforcement Learning algorithms include the learning curves for each experiment done and an analysis of their success rate and behavior. Each experiment was conducted based on the previously stated hypothesis. The results of each experiment are presented alongside their respective hypotheses, accompanied by considerations and analyses.

H1: *Does the RL algorithm outperform the greedy strategy in scenarios where the PIW can move out of the region of highest probability?*

Tests using RL algorithms on a 40x40 grid with dispersion increments of 0.1 and 0.05 show that RL outperforms the greedy approach in more complex environments. PPO significantly outperformed the greedy approach, with a 75.44% success rate versus 35.84% for 0.1 dispersion increment and 83% versus 50.18% for 0.05. RL algorithms can learn complex search patterns, coordinating to encircle the probability matrix in its front and rear intercepting the PIW, suitable for unpredictable SAR missions.

Figures in 2 and 3 display PPO learning curves for these configurations and compare PPO and DQN learning. According to these results, PPO learns a better policy much quicker than DQN, achieving rewards between 1.4 and 1.75, while DQN reaches only 0.4. PPO training took 3 hours, whereas DQN took 23 hours. PPO's advantage is attributed to its action probability distribution, allowing better exploration of states the agent has less knowledge about. Thus, PPO was chosen for further tests.

H2: *Does decentralized training improve convergence time and stability?*

To test this hypothesis, we trained a PPO algorithm using a decentralized approach, with each agent having its own policy rather than a shared policy among all

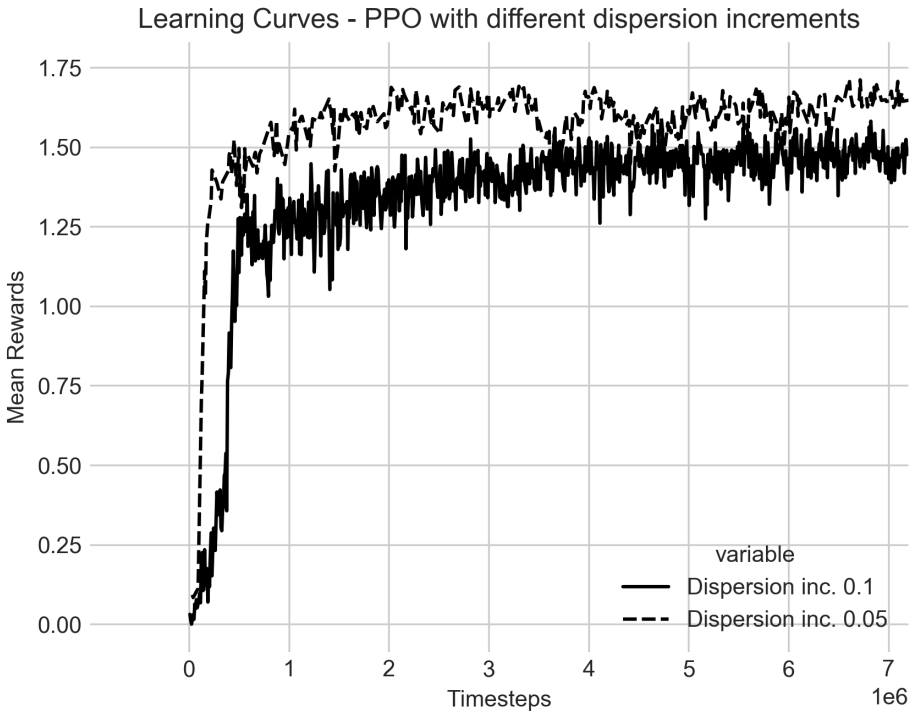


Figure 2. Learning curves for PPO on 0.1 and 0.05 dispersion increment configuration

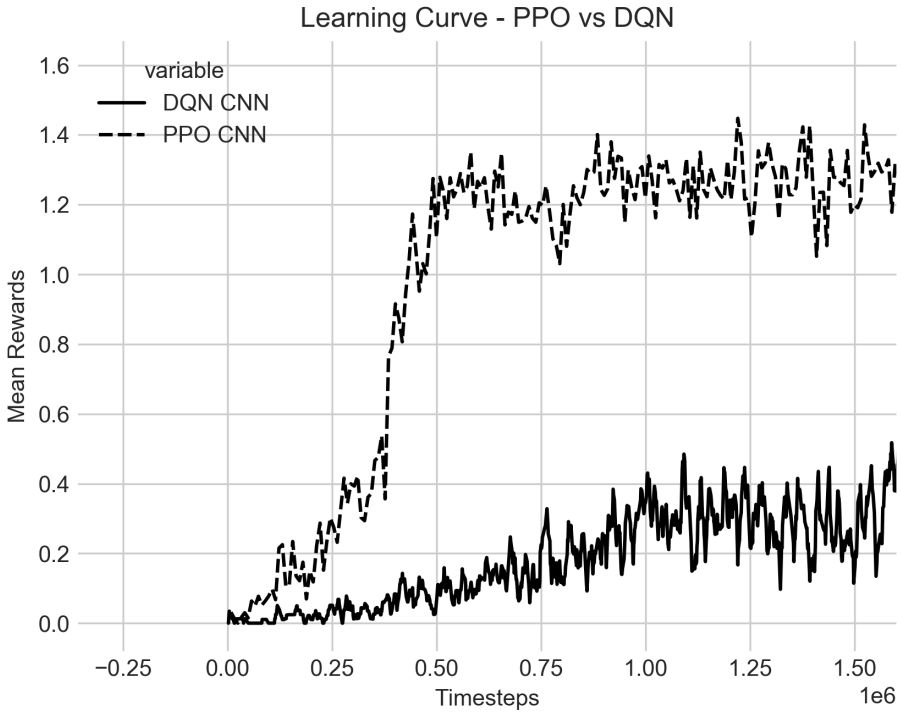


Figure 3. Learning curve comparison of PPO and DQN on 0.1 dispersion increment configuration

agents. This idea stemmed from mixed findings in recent literature, with some studies indicating improvements [Albrecht et al. 2024] in training time and stability, while others found no benefits or even detriments [Labanca 2024]. The training setup mirrored Hypothesis 1 with a 0.1 dispersion increment, differing only in using unique policies per agent. This allowed for direct comparisons with the standard PPO.

Decentralized training resulted in slower convergence, reduced stability, and inferior policy performance. Figure 4 compares centralized and decentralized PPO learning curves. Both centralized and decentralized approaches reached a mean reward of around 1.4, but the decentralized version took around two million steps compared to the centralized version's 500,000 to a million steps. Mean actions were around 40 for centralized and 50 for decentralized, indicating the decentralized approach was less efficient.

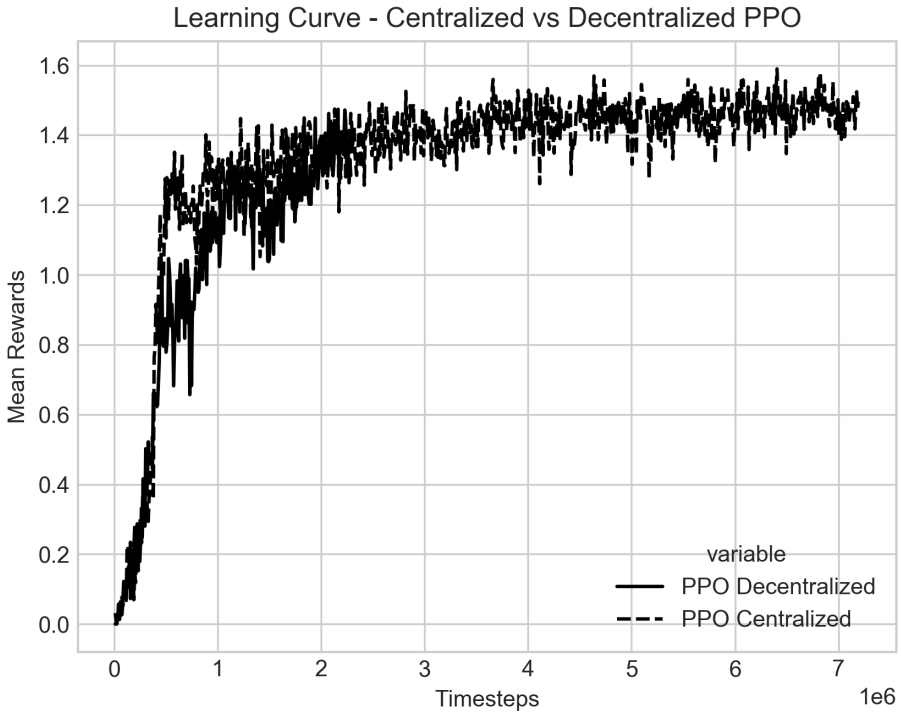


Figure 4. Learning curve comparison Centralized PPO vs Decentralized PPO.

The centralized approach consistently outperformed the decentralized one, with higher mean rewards (centralized = 1.34 and decentralized = 0.94), lower action median counts (centralized = 23 and decentralized = 41), and a higher success rate (centralized = 75.44% and decentralized = 53.22%). Consequently, we decided to use centralized training for all subsequent hypotheses.

The longer training times for decentralized training can be attributed to the additional resources required for multiple policies, each necessitating a separate neural network. The lower mean reward is likely due to some agents learning sub-optimal policies, significantly affecting overall performance in a complex problem with few agents.

H3: *Does search trajectory information impact search results?*

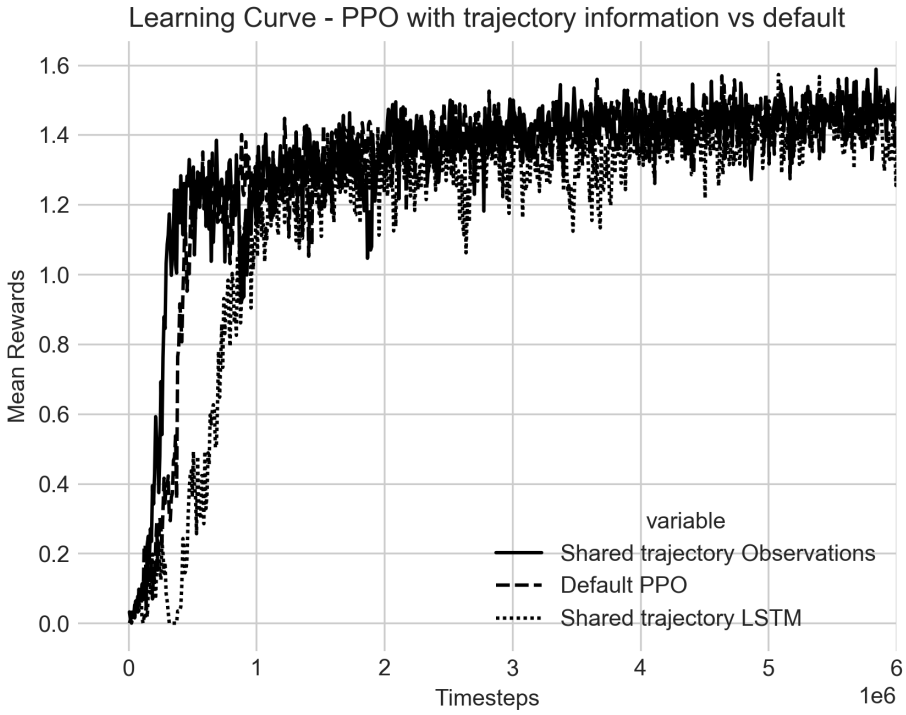


Figure 5. Learning curve comparison - PPO with trajectory information (matrix) vs PPO with LSTM vs default PPO.

For this hypothesis, we trained a centralized PPO algorithm using two approaches: (*i*) Trajectory-Based Probability Adjustment: modifying the probability matrix based on the drone's trajectory, reducing cell probability after a drone searches it and gradually restoring it over time, and; (*ii*) LSTM Layer Integration: adding a Long Short-Term Memory (LSTM) layer to handle the sequential information of agent positions, replacing the second fully connected layer in the positions branch of the neural network.

As seen in figure 5, the learning curves for PPO with trajectory information, both matrix and LSTM, are similar to the default PPO configuration. The results from the validation process, consisting of 5,000 simulations for each approach, also do not show any difference. The success rate of PPO default was 75.44%, the success rate of PPO with trajectory matrix was 75.98%, and the success rate of PPO with LSTM was 76.46%.

We hypothesize that the information in the environment's observation is sufficient for the algorithm's Markov Decision Process (MDP) to select optimal actions, rendering additional trajectory information redundant. It is speculated that trajectory information might be more efficient in larger-scale searches with more agents and greater search area overlap.

H4: *Do agents develop new search patterns when there is multiple PIW?*

This experiment differed from the others by including four PIWs instead of one to determine whether agents adapt their behavior to multiple PIWs. The learning curve in Figure 6 shows agents learning to coordinate their search as mean actions decrease

and mean rewards increase. The theoretical maximum reward is 8, but agents converged around 5 after approximately three and a half million steps, with mean actions oscillating around 80. This instability reflects the complexity of managing four distinct PIWs, each with its own movement pattern.

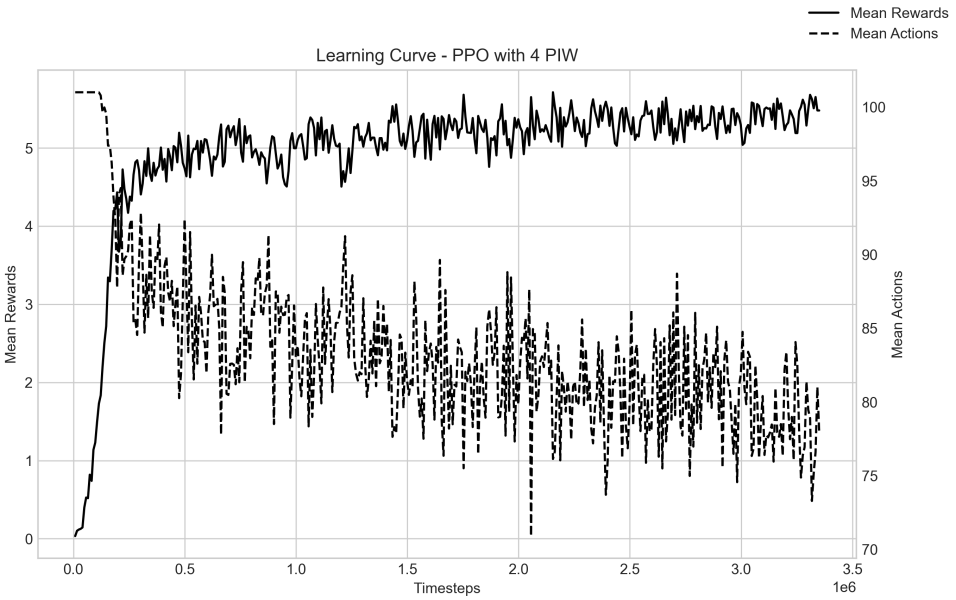


Figure 6. Learning curve for PPO with 4 PIW.

The agents found all PIWs in only 21.54% of the tests, a lower success rate than in other hypotheses. Further research and advanced RL features are needed to handle the increased complexity and stochastic nature of realistic environments. Although the agents found an average of 2.3 PIWs per test, achieving a 100% success rate is crucial for real-world scenarios where all PIWs must be rescued. While this result is a step forward in RL research, it highlights the need for ongoing improvement to ensure all PIWs are consistently located.

5. Conclusion

This paper presented an environment for the problem of Search and Rescue (SAR) shipwrecked people in the ocean and evaluated different RL algorithms for different scenarios in SAR. In the first hypothesis, we aimed to test if RL techniques have superior adaptability and performance compared to a pre-determined, informed algorithm, especially when PIWs deviate from regions of highest probability. The results confirm that RL algorithms outperform the greedy approach in all metrics. PPO achieved a 75.44% success rate with a 0.1 dispersion increment and 83% with a 0.05 increment, compared to the greedy algorithm's 35.84% and 50.18%, respectively. DQN failed to learn an effective strategy.

The second hypothesis examined the effect of independent networks on convergence time and training stability. Decentralized networks required more training steps and were less stable than centralized ones. This is due to the additional resources needed for multiple policies, each training its own neural network. The lower performance is likely

because not all agents learn an optimal policy, significantly impacting results in complex problems.

For the third hypothesis, we tested the impact of historical search data on search efficiency. The results showed no significant change in behavior or efficiency. We concluded that the environment's observations provide sufficient information for the algorithm to choose good actions. The additional data did not enhance performance, but it might be more impactful in missions with greater search area overlap.

Finally, the fourth hypothesis investigated agent organization and movement patterns in scenarios with multiple PIWs. Agents consistently located more than half of the PIWs, which is considered a success solely in research terms, since this still signifies loss of life in real-world applications. Although the agents didn't find all PIWs in most tests, they effectively coordinated their search, improving their performance over time. The increased complexity of managing multiple PIWs led to sub-optimal solutions, but the algorithms showed potential for handling such scenarios, to achieve an applicable 100% success rate more research is needed.

References

[Abreu et al. 2023] Abreu, L. D. M., Carrete, L. F. S., Castanares, M., Damiani, E. F., Brancalion, J. F., and Barth, F. J. (2023). Exploration and rescue of shipwreck survivors using reinforcement learning-empowered drone swarms. In *XXV Simpósio de Aplicações Operacionais em Áreas de Defesa*, pages 64–69.

[Ai et al. 2021] Ai, B., Jia, M., Xu, H., Xu, J., Wen, Z., Li, B., and Zhang, D. (2021). Coverage path planning for maritime search and rescue using reinforcement learning. *Ocean Engineering*, 241:110098.

[Albrecht et al. 2024] Albrecht, S. V., Christianos, F., and Schäfer, L. (2024). *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

[Alotaibi et al. 2019] Alotaibi, E. T., Alqefari, S. S., and Koubaa, A. (2019). Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832.

[Falcão et al. 2024] Falcão, R. L., de Oliveira, J. C. C., Andrade, P. H. B. A., Rodrigues, R. R., Barth, F. J., and Brancalion, J. F. B. (2024). DSSE: An environment for simulation of reinforcement learning-empowered drone swarm maritime search and rescue missions. *Journal of Open Source Software*, 9(99):6746.

[Labanca 2024] Labanca, A. M. (2024). Deep reinforcement learning for kamikaze drone decision-making. Master's thesis, Instituto Tecnológico de Aeronáutica.

[Li et al. 2015] Li, X., Li, L., Gao, J., He, X., Chen, J., Deng, L., and He, J. (2015). Recurrent reinforcement learning: A hybrid approach. *arXiv preprint arXiv:1509.03044*.

[Liang et al. 2018] Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I. (2018). Rllib: Abstractions for distributed reinforcement learning. International Conference on Machine Learning.

[Mnih et al. 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–533.

[Schuldt and Kurucar 2016] Schuldt, D. W. and Kurucar, J. A. (2016). Maritime search

and rescue via multiple coordinated uas. Technical report, Defense Technical Information Center.

[Schulman et al. 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, abs/1707.06347.

[Silver et al. 2021] Silver, D., Singh, S., Precup, D., and Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, 299:103535.

[Wu et al. 2024] Wu, J., Cheng, L., Chu, S., and Song, Y. (2024). An autonomous coverage path planning algorithm for maritime search and rescue of persons-in-water based on deep reinforcement learning. *Ocean Engineering*, 291:116403.