ISSN: 1983 7402 ITA, 26 a 28 SET 2023

Exploration and Rescue of Shipwreck Survivors using Reinforcement Learning-Empowered Drone Swarms

Leonardo D. M. de Abreu[1], Luis F. S. Carrete[1], Manuel Castanares[1], Enrico F. Damiani[1], José Fernando B. Brancalion[2], Fabrício J. Barth[1]
[1]Insper, São Paulo/SP - Brasil
[2]Embraer, São José dos Campos/SP - Brasil

Abstract— **The goal of this project is to create a reinforcement learning algorithm that locates shipwrecked individuals using a swarm of drones. A simulated environment was developed to train and visualize the outcome of the trained algorithm considering the ocean's dynamic circumstances. This project does not discuss image recognition of shipwrecked people, since the true focus of this project is to optimize the search routine of a drone to find the target in the most efficient way possible. The implemented Reinforce algorithm takes into account a dynamic map of probabilities, representing the chances of a person being found, as well as the position of other agents. Outcomes include an open-source Python package for the environment and the implementation of the reinforcement learning algorithm. The algorithm demonstrates superiority over the predefined approach, proving the advantages of reinforcement learning in efficiency and effectiveness.**

Keywords— **Multi-Agent Systems, Reinforcement Learning, Simulation**

I. Introduction

Every year, vast bodies of water worldwide claim numerous missing individuals. According to the World Health Organization (WHO), there are an estimated 236,000 annual drowning deaths worldwide, making it the third leading cause of unintentional injury/death worldwide and accounting for 7% of all injury-related deaths [1]. With over 71% of the earth's surface covered by oceans[2], finding these missing individuals is no easy task, due to the complexity of oceanic environments and the vastness of the search areas. However, drone swarms have emerged as a promising tool for searching for missing individuals.

The use of drones in rescue operations has resulted in successfully saving 940 people while being utilized in 551 rescue incidents so far [3]. The capacity of drones to reach difficult terrain and inaccessible areas, as well as their ability to capture real-time images and videos, has proved to be helpful in search and rescue missions.

The success rate of search and rescue missions is believed to be significantly increased by the incorporation of Artificial Intelligence (AI) technology [4], as it can leverage probabilistic models based on the ocean's behaviors, as well as the last known location of the people being rescued. Several solutions have been proposed in the last years to solve this problem [5], [6], [7], in special, using reinforcement learning algorithms. By utilizing AI and Reinforcement Learning (RL), the project aims to investigate algorithms that can improve the effectiveness of search and rescue operations in dynamic environments, ultimately resulting in more lives saved [5], [6], [7].

This paper focuses on the demonstration and explanation of two distinct search algorithms. An algorithm that uses predefined paths to search for the target, and a multi-agent reinforcement learning algorithm, which is expected to learn and optimize the search autonomously. The objective is to create an algorithm that is capable of finding a shipwrecked person in the most efficient way possible. Furthermore, the environment that will be developed will replicate the circumstances of the ocean simply. To do so, a scenario will be created, where the environment changes dynamically, to update the probabilities in which the target could be. This does not require an accurate representation of how the ocean acts. Moreover, since the project deals strictly with simulated environments, image recognition, and real-life testing will not be considered. However, parameters regarding real-life issues are explored, for example, the number of drones needed and their technical limitations, as well as different deployment strategies.

II. Literature review

To improve the performance of Unmanned Aerial Vehicles (UAVs) in Search and Rescue (SAR) missions, Alotaibi [5] proposes the Layered SAR (LSAR) algorithm. The key idea is, that in disasters, there is a location where most of the possible survivors are located. Therefore, the algorithm initially, focuses the search on this specific location and gradually moves away.

This technique uses a single command base, utilizing a cloud server as the drone mission controller. This cloud server is implemented through a cloud-based management system that is used to facilitate communication and collaboration among a swarm of drones, as well as control UAVs and schedule their missions.

While searching for survivors, if a person is found, the UAV records his location. Concurrently, the searcher UAV periodically reviews the locations to determine whether it meets a predefined threshold. When this threshold is met, the searcher UAV aircraft signals the cloud server for assistance. Following this, the server initiates an inward shift, beginning from the layer where the request for help originated. Once a drone sends a request for help, it is reassigned as a rescuer. Meanwhile, the remaining aircraft that underwent layer shifting continue their search activities as searchers in their newly assigned layers. When the UAV rescuer completes its mission, the other drones begin to return to their original layers, and the process starts anew. In situations where many drones are allocated to the same layer, the process is similar, except that only one drone of the layer becomes a rescuer.

Another paper that delved into the utilization of UAVs for SAR missions is *Maritime Search and Rescue via Multiple Coordinated UAS* [7]. In it, there is the development of an Unmanned Aerial System (UAS) focused on the search element of these missions. The article devised a strategy that uses multiple UAS in cooperation. This solution begins by selecting an area, covered by a probability distribution, and then partitioning it. After that, each drone is assigned a partition, and it is ensured that a minimum distance is maintained between each drone. Also, a greedy algorithm pairs the most capable UASs with the most significant partitions. The capability of a UAS is determined based on the quality of its camera and its endurance, while the significance of a partition is related to the chances of having a survivor. Path evaluation follows those steps, in which path and altitude are assigned to each drone. There are four possible pre-defined paths, devised by the article. In addition, this article also developed an equation that can calculate the visible area of the drone at a given time.

Now looking at reinforcement learning (RL) solutions, Jia [6] 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. In this algorithm, the next action of the agent is based on its current state. Thus, given the current state of the agent, the action with the highest reward from the previous training is selected from the Q-table.

The experiment's comparative results indicate that the search path generated by the model can safely cover the entire SAR area, maintaining a low rate of repetitive coverage, and preferentially searching high-probability areas. However, there are limitations to this study. The "search area is assumed to be constant" during the path-planning process and the model does not consider the impact of wind, waves, and moving obstacles.

In terms of similarities, both [5] and [6] employ probability regions to locate the target. Nevertheless, while the LSAR approach always places the highest probability region at the center, [6] calculates the probability based on maritime data and statistical methods. In addition, the [7] and [5] utilize multiple agents to complete the SAR mission faster. However, while in [5] the drones just communicate with each other when individuals are found, in [7] there is no communication between the agents. Although all three approaches aim to optimize SAR missions and increase the success rate, they utilize distinct strategies to achieve these goals.

III. Simulation tool and environment

To achieve what was proposed, a simulation of the real-world situation had to be created, but since the real world is incredibly complex a few premises had to be set to simplify the overall scenario. First of all, there will only be one shipwrecked person. Additionally, it is considered that once the drone is over the person, and executes the action of search, he will identify the person. How he identifies the person is not considered, since this is only a simulated environment. The drone will be able to move only in the area delimited by a grid that covers where the shipwrecked person is located. The drone can only execute five different actions, moving up, down, left, right, and searching. The search was defined as an action because, in this scenario, the drone will be flying at a high altitude so that it can visualize a bigger area. Once it identifies a possible target it will descend and verify that it is in fact a person, therefore the search action was included to represent this process. The drone also does not move diagonally to simplify the model. The environment will not simulate the wind or any natural disaster which may affect the drone's flight.

The drones have a restriction, their battery life. They can only do a certain number of steps before their battery ends. The person's movement in the ocean will also not be defined by complex ocean modeling but by a simple vector that will force the person to drift away over time. Finally, the drone will be placed in a grid, where each cell has a probability, representing the chances of the shipwrecked person being located in that area.

Based on previous studies [6], [7] a probability matrix will be created to demonstrate the chances of the shipwrecked person being in a given cell. The matrix has the same dimensions as the position matrix, and it is the primary piece of information used by the agent. Researchers with similar areas of study [6], [7] used multiple metrics in order to define the values of the matrix. For example, the wind and flow of the ocean can greatly impact the trajectory of a shipwrecked person. This type of data can vary depending on the place, day, and time. However, since the modeling of the ocean is not a priority of the project, a directional vector will act as the ocean's current, which will subsequently drag the shipwrecked person to different places on the map. This will, therefore, change the value of each cell in the probability matrix. Said directional vector along with the initial position of the shipwrecked person are inputs that can be defined by the user. This allows the simulation of a scenario in which the user has knowledge of the ocean's current, along with the shipwrecked person's last known localization.

Since the goal of this library is not to simulate in depth the ocean's movements or the person's movement in the ocean, the person's movement in the grid will be created using a probability matrix which is described above. In the simulation, the person will start in a cell chosen by the user where the probability of the person being there is 100%. In the next step, the probability will disperse, using a Gaussian distribution. Considering the dispersed probability matrix the person will look at all the adjacent cell's probabilities and will make a decision either to move or to stay in its current spot, according to the probability of the adjacent cells.

This movement strategy was adapted to simulate the target's decision-making. When searching for a person in the ocean, it is doubtful they will stay in the same place, but instead, they will constantly be trying to make decisions to survive, meaning, they would most likely move around, while being dislocated by the current. Although in a real situation, a shipwrecked person may not move as fast as the target in the simulation, the movement is also designed to simulate the uncertainty of a person being in a cell. Even though the person may not be in a high-probability cell, the agent still must search that area, because the person will most probably be located in one of the other high-probability cells. This type of behavior, by the agent, is reflected in the environment's rewards.

The reward is a simple concept where you will penalize the agent if it does something that it is not supposed to do and reward it in case it does something that leads it to its goal. Any reinforcement algorithm works in such a way that it will always try to maximize the agent's rewards, so if the agent does something it is not supposed to do, it will receive a massive negative reward so it learns to not do it again.

In this environment, the agent receives a reward of 1 per action by default, because the drone needs to be incentivized to walk and explore the grid. The drone (agent) will receive a reward of -100000 in case it leaves the grid. This is because in the early experiments when the reward for leaving the grid was -1000, the agent would learn that leaving the grid instantly would give a better reward than searching and not finding the target since if he left the grid the reward would be only -1000. Alternatively, if the agent searched and in the end did not find the person, the reward would be about -2000. Therefore the reward for leaving the grid was raised to -100000 so that the agent quickly learns to not leave the grid.

The agent will also receive a reward of -1000 if it does not find the target. This is so that the agent is penalized for not finding the target, but not as much as leaving the grid since the agent must still be incentivized to look for the target. The agent will receive a reward of -2000 in case of collision because the reward needs to be lower than the case in which the agent does not find the target, otherwise, the agents would learn to crash so that they don't get a worse reward.

In case the agent searches, it will receive a reward according to the probability of the cell, so if the drone searches in a cell with a probability of 80% the drone will receive a reward equal to 8000, this is because the agent needs to learn that it is better to search in higher probability cells rather than waste time searching in the lower probability areas. This is always true, except in the case that the probability in the cell is lower than 1%. In this case, the reward is -100, to disincentivize the agent to search in low-probability cells.

Finally, if the drone finds the target it will receive a reward according to (1). This is because the agent needs to be incentivized to find the target in the fewest moves possible. So if the total timestep is 500 and it finds the target in the timestep 480, it will receive a reward of 200. Still, if the person is found in 100 steps, it will receive a reward of 4000, greatly incentivizing him to find the target in the quickest way possible.

$$r = 10000 + 10000 * (\frac{1 - timestep}{timestep\_limit}) \quad (1)$$

Where $timestep$ represents the number count of the action that is taking place. For example, if an agent executed 50 actions, the timestep is equal to 50. $timestep\_limit$ is the number of actions that an agent can do in an episode. An episode refers to a series of actions an agent takes from its starting state to its conclusion.

This environment was implemented as a python library [8], [9], that follows the norms of PETTINGZOO project [10], with the intention of making it available for future studies. The source code for this package is also publicly available on [11]. This repository contains detailed documentation of the environment, including installation instructions, thorough descriptions of functions and variables, and an example of how this environment is to be used.

IV. ALGORITHMS

After developing the environment, a baseline algorithm was created to be used as a reference point for performance comparison, to evaluate whether the reinforcement learning strategy is improving the search effectiveness or not. This baseline algorithm utilizes multiple drones with pre-defined search patterns movements. This strategy has been used in other search and mapping papers, making it suitable for comparison [7], [12].

Among all the pre-defined search patterns, the Parallel Sweep Search was chosen as it is usually used in situations when the search object location is uncertain, and the search area is large, such as a flat terrain or water, where the area needs to be divided into sub-areas and assigned to individual search facilities on-scene at the same time [12].

The Parallel Sweep Search involves traversing a rectangular search area using parallel movements until the search object is found, as shown in Fig. 1. If the target is not found and the agent reaches its last position, the drone starts the Parallel Sweep Search in the opposite direction backtracking its steps.

To simplify the implementation, the baseline algorithm was implemented to work only with 1, 2, and 4 drones. When employing 2 or 4 agents, in order to ensure an equal division of the terrain, only matrices with an even size can be utilized. Using just one agent, the drone originates from the top-left corner of the search area, while using 2 agents, one drone starts from the origin, and the other begins from the opposite cell, situated at the bottom-right corner. With four agents, each drone is positioned at one of the corners of the matrix. After the drones are positioned, they conduct parallel scans toward the center of the matrix. This strategic approach aims to encircle the target, thereby enhancing the likelihood of the drones successfully locating it. Fig. 1 illustrates this strategy.

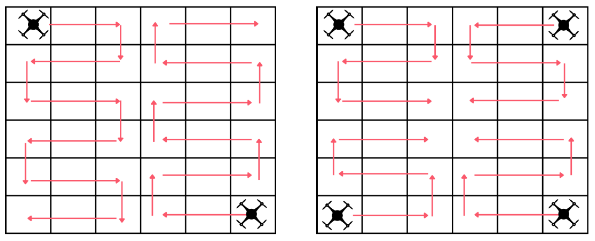
Fig. 1. Parallel Sweep representation using 2, and 4 drones

This algorithm has some limitations, such as not considering the probability matrix and requiring the drone to perform a search action for each cell in the matrix, increasing the number of actions needed. Moreover, the algorithm relies on an equal division of the search area among the drones, making it impossible to utilize an odd number of drones. For instance, if there are 5 drones available, only 4 can be effectively deployed, while 1 would remain idle.

A. *Reinforcement Learning implementation*

In this work, we used a policy-based method. Policy-based methods search directly for the optimal policy by finding the best policy parameters that map states to actions, aiming to maximize the cumulative reward [13].

We implemented the Reinforce algorithm [14] in this work due to its simplicity compared to other methods, serving as a starting point for investigating connectionist reinforcement learning. The Reinforce algorithm [14] is stochastic and incorporates randomness during training and decision-making. It utilizes a neural network's output, which represents a probability distribution of actions, to select the best action for a given state. Different actions can be chosen for the same state.

One of the main benefits of this approach is its smooth representation, where slight adjustments to the network weights result in only small changes to the neural network's output. This is in contrast to Value-Based methods, where small weight changes can lead to different actions [13].

The policy gradient strategy aims to reinforce favorable actions by increasing the probabilities of actions with higher returns and decreasing the probabilities of actions with lower returns. This is achieved by receiving rewards from interactions with the environment. The process continues until an optimal policy is achieved [13].

The implementation of the Reinforce algorithm was made with the PyTorch Python library [15]. This library is an optimized tool for implementing tensors, a multidimensional array of numbers, for Deep Learning. Furthermore, different training configurations were determined. More specifically, three different configurations were created, for one agent, two agents, and four agents. All the configurations were based on a 20×20 grid and a time limit of 100. The paper [9] explains why this grid size is a reasonable size for this type of simulation.

The chosen approach was to use a single Neural Network for all the drones. This decision was based on the concept that all agents have the same objective, finding the shipwrecked person, as fast as possible, without colliding and leaving the grid. Therefore, regardless of the drone, given a situation and the position of the other agents, the chosen action should be the same.

The Neural Network is composed of three fully connected layers, the first is the input, the second is the hidden, and the third is the output. The first layer receives the position of the Drone making the action (one neuron received the X position in the matrix, and another received the Y position), the position of the other agents (each represented by 2 neurons, for the X and Y values), as well as the ten higher probability cell locations, aiming to decrease the number of episodes needed to train the neural network, comparing to the entire matrix.

In the first layer, still, the ReLU (Rectified Linear Unit) activation function was chosen. The hidden layer was composed of 512 neurons, and the activation function was also the ReLU. Finally, the output layer was composed of all the possible actions of the drone, totaling 5 neurons (one for each action). As each action is represented by a neuron, and only one action should be chosen, the Softmax (also known as the normalized exponential function) activation function was used for the layer, as it returns the probability of each action being the one to be picked. Also, another hidden layer was introduced, with 256 neurons and the same ReLu activation function, enabling more weights to be set, consequently, improving the performance of the algorithm.

Therefore, based on the above Neural Network, a Reinforce algorithm was developed. In this algorithm, for each agent in each step of an episode, an input tensor was generated and given as input for the Neural Network. Due to the stochastic characteristic of the algorithm, an action was chosen based on its probability (the choice, itself, was based on it). Then, the actions are passed to the environment, and as a consequence, a reward for each agent is received. Having the rewards, actions, and states, it is calculated the loss for each agent in each step. This loss is then used to update the weights in the Neural Network (trying to minimize it). The training stops when a total of 300000 episodes is reached, or more than 80 consecutive results above 100000 total rewards are achieved.

V. Results

To measure the success of the algorithm, multiple tests were executed after the training phase of the neural network, which took into account different numbers of drones. A neural network was created and trained for a single drone, two drones, and four drones. After these three neural networks were trained, they were each tested, with the objective of analyzing two specific results.

The initial analysis focused on the collected training data for all configurations. It revealed the cumulative sum of rewards and the number of actions taken within each 500 episode interval, as shown in Fig. 2, Fig. 3, and Fig. 4.

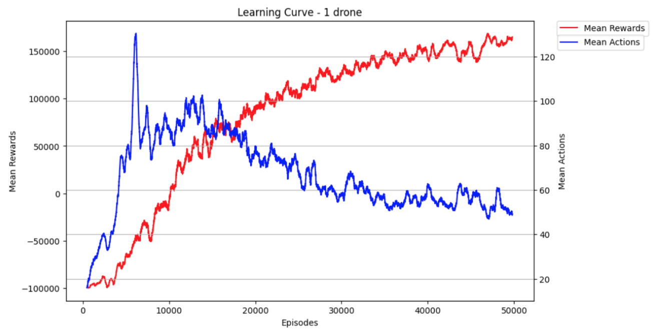

Fig. 2. Learning Curve for Configuration 1, using one drone for a 20x20 matrix

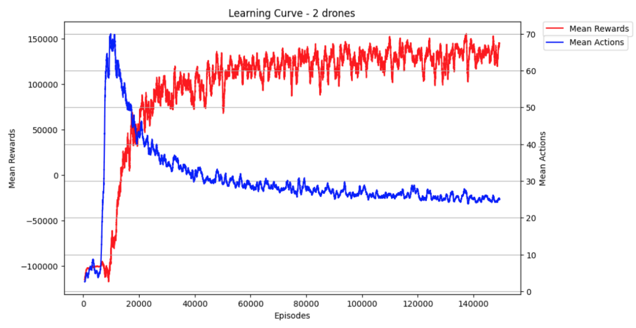

Fig. 3. Learning Curve for Configuration 2, using two drones for a 20x20 matrix

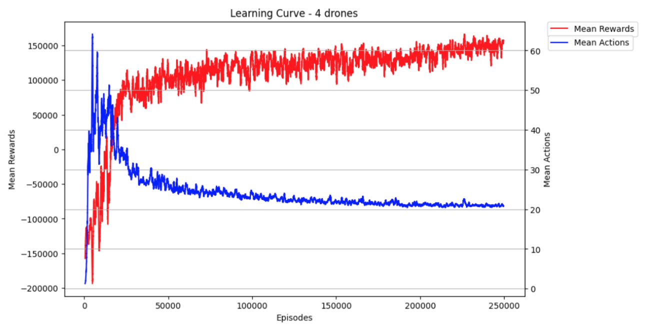

Fig. 4. Learning Curve for Configuration 3, using four drones for a 20x20 matrix

A similar pattern was observed across all configurations, indicating a consistent learning trend. Initially, the agents started with low values, gradually increasing over time. This indicates that the agents learned from their experiences, gradually improving their performance.

The learning curves, combined with the number of actions, revealed three distinct stages of learning. In the initial stage, the agents learned to avoid invalid actions, such as moving outside the grid. Subsequently, they focused on exploring the environment, as evidenced by a significant increase in the number of actions. Finally, the agents entered an optimization phase where they aimed to find the target as quickly as possible while minimizing the number of actions.

It is also possible to notice that the agents recognized the correlation between fewer actions and higher rewards. This observation explains the decreasing trend in the number of actions over time. The drones learned to optimize their actions to maximize their overall reward.

By analyzing the three learning curves, it became evident that the convergence time increased with a higher number of agents. This outcome was expected since more agents need to learn optimal actions that maximize their chances of finding the target while minimizing collisions with other drones. On the other hand, it can be inferred that the utilization of additional drones leads to a substantial reduction in the number of actions. This inference is supported by the observable decline in the number of actions illustrated in the transition from Fig. 2 to Fig. 3, as well as from Fig. 3 to Fig. 4.

After conducting the learning analysis, comparison tests were performed on the implementations of the Reinforce and Parallel Sweep algorithms. For this purpose, experiments were conducted for the three presented configurations.

The comparison metrics employed included the frequency of successful target identification, serving as an indicator of the model's performance, and the number of actions executed, which measures the time taken by the drones to locate the target, an important metric given the time-sensitive nature of the problem.

Each algorithm was tested 10000 times for each configuration to generate the experimental results. In these tests, the speed of the castaway assumed values of -0.1, 0, or 1 for both the x and y axes, chosen randomly in each test, in order to measure the efficiency of the algorithms for different directions and orientations that the castaway could assume. It is important to note that the drones always started at their initial positions mentioned earlier for each configuration, and the castaway always started in the middle.

For Configuration 1, Fig. 5 shows a boxplot of the number of movements the drone takes to locate the target. It can be observed that the drone takes an average of 46.66 actions, with a median value of 37 actions to find the person, with a success rate of 88.2%, while the baseline only achieves 5.1%, as shown in Fig. 6. This discrepancy for this configuration is largely due to the limited number of actions within 100 steps, and the drone does not have enough battery to locate the target. This explains why the average number of actions for the Parallel Sweep algorithm is 100, as they reached the maximum number of actions possible without success.

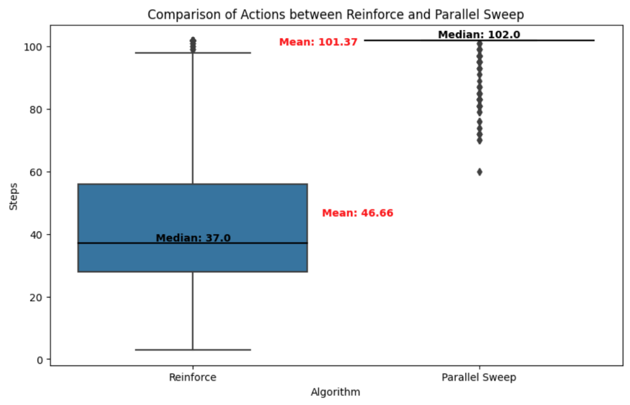

Fig. 5. The distribution of actions quantity per episode, considering configuration 1

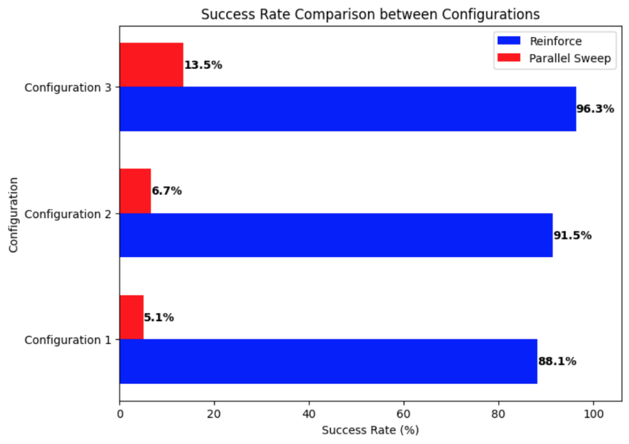

Fig. 6. Success rate comparison using Reinforce and Parallel Sweep for all configurations

VI. Future Work

Some further exploration and improvements can be addressed for future iterations. Regarding the environment, one of the biggest limitations is the fact that the shipwrecked person will not leave the grid if it reaches its edge, but will simply move around in the corner until the episode is complete. Second, the drones can not move diagonally, which does not represent the real world, where the drone is free to move in any direction. Finally, there is also a limitation with the ocean's simulation and the target's movements. Although it was sufficient for this project, it may not be for a real-life situation, so it would be interesting to add a more sophisticated way to calculate the probability matrix as well as a more complex simulation for the target's movement.

Now regarding the algorithm, there are a few improvements that can occur. Firstly, it would be interesting to investigate if different hyperparameters can fine-tune the model and potentially achieve better performance. Secondly, exploring if different network architectures and activation functions can enhance outcomes. The choice of architecture and activation function significantly influences the model's performance, and evaluating alternative options can provide valuable insights. Additionally, it is important to consider and compare different reinforcement learning algorithms, investigating their behaviors, step efficiency, and the success rate of each model. Furthermore, comparing the developed model with other baselines, such as greedy search, can lead to profound discoveries about its relative performance. This analysis will help identify the strengths and weaknesses of the model and highlight areas for improvement.

Although the current implementation focuses on a 20x20 matrix, which corresponds to a search area of 3.6 km², future work can train the algorithm on matrices of varying dimensions, in order to ensure that the solution is adaptable to different search area sizes and can be applied to a wide range of scenarios. As the search area increases, it is worth investigating the necessity of applying drone relays, to ensure uninterrupted communication during drone missions for wide areas.

Exploring new drone parameters, such as sensors, can possibly enhance their capabilities, as they can provide data for various applications, including environmental monitoring and infrastructure inspections. Moreover, creating defense mechanisms when drones approach the end of their battery life is also a possible future enhancement. Implementing fail-safe mechanisms, such as automatic return-to-home protocols or emergency landing procedures, can help prevent accidents or potential damage caused by drones falling due to depleted batteries.

VII. Conclusions

The project yielded two distinct outcomes: the development of an environment and the implementation of the reinforcement learning algorithm. The environment, presented as a Python package, was designed with the intention of allowing external researchers to utilize and modify it as needed. This open approach encourages others to build upon the project, potentially achieving even more remarkable results than those demonstrated here. By engaging a wider community, the utilization of this environment has the potential to drive further improvements, thereby influencing future algorithmic advancements.

The algorithm itself, showcased its effectiveness and superiority in solving the given problem when compared to the predefined parallel sweep approach. The findings provided proof of concept, supporting the notion that reinforcement learning strategies outperform predefined paths in terms of efficiency and effectiveness.

Furthermore, the agents trained using this algorithm consistently demonstrated the ability to locate the target swiftly across various configurations. This aspect aligns with the critical need for prompt responses in scenarios involving shipwrecked individuals. Moreover, scaling up the number of drones resulted in faster and more accurate responses, justifying the utilization of multiple agents. However, as previously mentioned, there are still some enhancements that can be explored in order to refine the solution's performance. Additionally, new scenarios and parameters can also be explored.

The comprehensive research and implementations carried out in this project were driven by the desire to advance search and rescue techniques. As emphasized in the introduction, the search for individuals lost at sea is both complex and of utmost importance. By continually exploring and refining new strategies, search and rescue missions can gradually enhance their effectiveness, ultimately resulting in more lives being saved.

References

[1] W. H. Organization, "Drowning fact sheets," 2021, accessed in: Feb. 22, 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/drowning

[2] U. S. G. Survey, "The distribution of water on, in, and above the earth," 2019, accessed in: Feb. 22, 2023. [Online]. Available: https://www.usgs.gov/media/images/distribution-water-and-above-earth

[3] DJI, "Drone rescues around the world," 2023, accessed in: Feb. 25, 2023. [Online]. Available: https://enterprise.dji.com/drone-rescue-map/

[4] E. Eliaçık, "Artificial intelligence vs. human intelligence: Can a game-changing technology play the game?" 2022, accessed in: Feb. 25, 2023. [Online]. Available: https://dataconomy.com/2022/04/is-artificial-intelligence-better-than-human-intelligence/

[5] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "Lsar: Multi-uav collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019.

[6] 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," *Ocean Engineering*, vol. 241, p. 110098, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801821014220

[7] D. Schuldt and J. Kurucar, "Maritime search and rescue via multiple coordinated uas," U.S. Department of Defense, Defense Technical Information Center, Tech. Rep. AD1030377, January 2017. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1035043.pdf

[8] L. F. Carrete, M. Castanares, E. Damiani, and L. Malta, "Dsse: Drone swarm search environment," 2023. [Online]. Available: https://pypi.org/project/DSSE/

[9] 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," 2023.

[10] F. Project, "Pettingzoo," https://pettingzoo.farama.org/, 2023, accessed in: Mar. 23, 2023.

[11] L. F. Carrete, M. Castanares, E. Damiani, and L. Malta, "Dsse source code," 2023. [Online]. Available: https://github.com/PFE-Embraer/drone-swarm-search

[12] K. CHOUTRI, M. LAGHA, and L. DALA, "A fully autonomous search and rescue system using quadrotor uav," *International Journal of Computing and Digital Systems*, vol. 10, pp. 2–12, 2021.

[13] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019. [Online]. Available: http://arxiv.org/abs/1912.01703