Appling Coverage Path Planning Algorithms in Search and Rescue Operations

Insper
Artificial Intelligence and Robotics

1 Introduction

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

According to Wu et al. (2024), searching for PIWs includes three main tasks:

1. accurately and quickly predicting the drift trajectory of PIWs;
2. determining the optimal search area to ensure full coverage of the possible distribution range, and;
3. planning the search path for the SAR units and maximizing the cumulative probability of success (POS) of the entire search process.

The first task is usually solved using drift prediction models, while the second and third tasks are solved by coverage path planning (CPP) algorithms. The usual algorithms applied to coverage path planning activities are Parallel Sweep Search, Parallel Line Scanning, Expanding Square Search, and A* algorithm.

This project aims to analyze different CPP algorithms that can be used in SAR operations. The algorithms should be able to determine the optimal search area and plan the search path for the SAR units, maximizing the cumulative probability of success of the entire search process. Each group will use the **Drone Swarm Search Environment**[1] Falcão et al. (2024) to evaluate the algorithms cited above and implement a list of CPP algorithms.

2 Drone Swarm Search Environment

The environment is a simulation tool to train reinforcement learning agents for search and rescue operations in maritime scenarios. However, this environment can also be used to evaluate conventional CPP algorithms. The environment is a 2D grid (figure 1) with a matrix representing the probability of containing a PIW. This environment uses maritime and wind satellite data to simulate the drift of PIWs from a specific point (latitude and longitude) from a specific date and time. The drift of PIWs is implemented by **OpenDrift**[2] Dagestad et al. (2018). The environment receives a list of latitude and longitude points for each particle from OpenDrift and then applies some mathematical transformations to create a probability matrix. The environment could generate a completely different probability matrix depending on the date, time, position, and hours informed. For example, the following setup was executed on September 23, 2024:

```
from DSSE import CoverageDroneSwarmSearch
env = CoverageDroneSwarmSearch(
    disaster_position=(-24.04, -46.17),  # (lat, long) some place near Guaruj
```

[1] https://pfeinsper.github.io/drone-swarm-search/
[2] https://github.com/OpenDrift/opendrift

```
    pre_render_time=10, # hours to simulate
)
env.save_matrix('../data/config_01.npy')
```

generated the environment shown in figure 1.

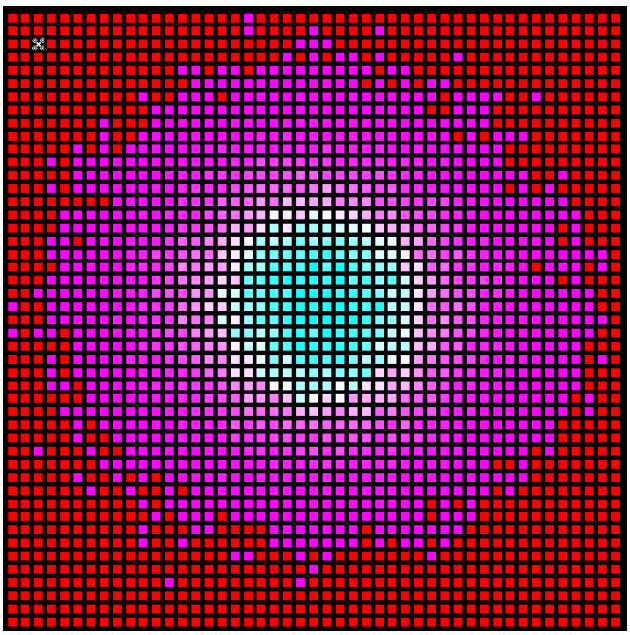


Figure 1: Visual representation of an environment

Different simulations can be executed after generating the matrix representation of the probability of containing a PIW. The code in listing 1 creates a simulation with two random agents.

```
from DSSE import CoverageDroneSwarmSearch

env = CoverageDroneSwarmSearch(
    drone_amount=2,
    render_mode="human",
    prob_matrix_path='../data/config_01.npy',
    timestep_limit=300
)

opt = {
    "drones_positions": [(20, 20),(10,10)],
}

def random_policy(obs, agents):
    actions = {}
    for agent in agents:
        actions[agent] = env.action_space(agent).sample()
    return actions

observations, info = env.reset(options=opt)

step = 0
while env.agents:
    step += 1
    actions = random_policy(observations, env.agents)
```

```
    observations, rewards, terminations, truncations, infos = env.step(actions)
    print(infos)
```

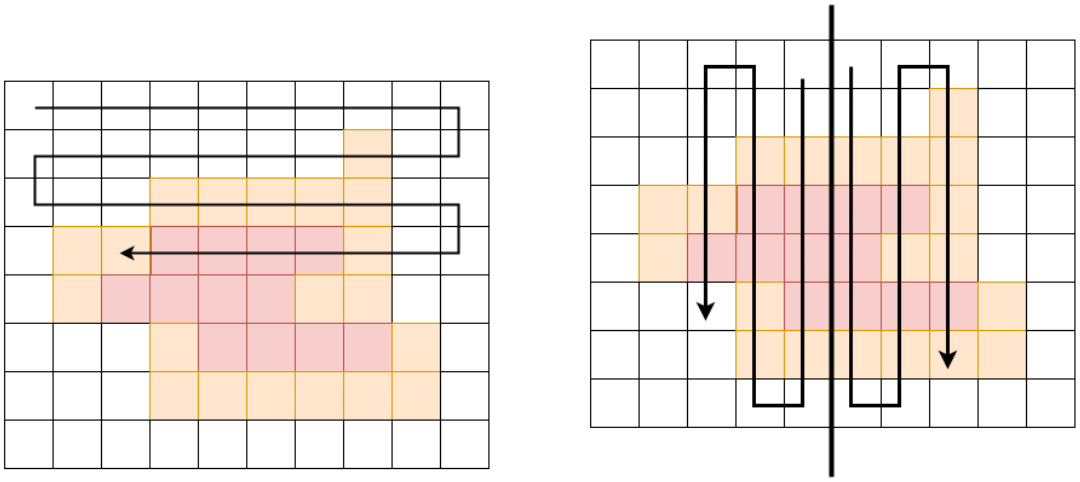
Listing 1: Simulation example

More information about how to use the DSSE package is available on https://pfeinsper.github.io/drone-swarm-search/Documentation/docsCoverage.html#about.

3 Coverage Path Planning algorithms

When planning for a marine SAR operation, a regular search mode is often selected to cover the search area. The result of the planning is fixed, which lacks the ability of autonomous obstacle avoidance and flexibility. The figure 2a shows the traditional maritime search pattern.

The same pattern can be applied with more than one drone (agent), like presented in figure 2b with 2 agents and in the figure 3a with 4 agents. The traditional maritime search pattern with one agent and with two agents are used to search a large area when the location of a survivor is uncertain. The search begins at one of the corners of the search area, and the orientation is generally in the direction of the search object's drift.



(a) Traditional maritime search pattern

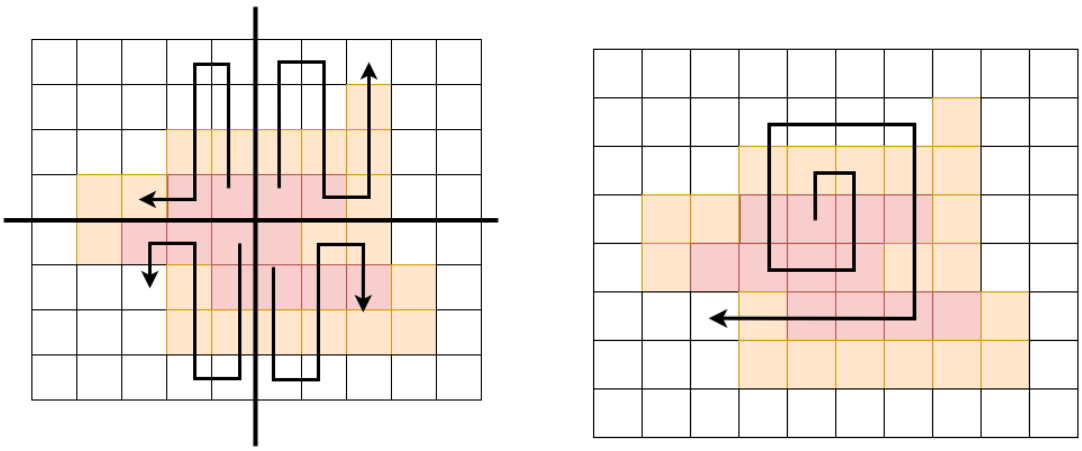
(b) Traditional maritime search pattern with 2 agents

Figure 2: Tradicional maritime search patterns

It is possible to split the area into four and apply the traditional maritime search pattern, like presented in the figure 3a. However, in order to apply this pattern with four agents it is necessary to know where is the center of region of interest, otherwise is it not possible to share the region in four equal parts.

In situations where we know the center of region of interest. In other words, when we know the region of highest probability of containing a PIW then we are able to apply a Expanding Square Search algorithm. The idea of this algorithm is to prioritize the area closest to the start and then gradually expand the search outward in a square pattern, like presented in the figure 3b.

Besides all the algorithms presented, informed search algorithms like A^* can be used.



(a) Traditional maritime search pattern with 4 agents

(b) Expanding Square Search

Figure 3: Maritime search patterns

4 Method and Expected Results

Each group must compare:

- tradicional maritime search, expanding square search, and A^* search in two different scenarios (i.e., the different matrix representing the probability of containing a PIW) considering only one agent;
- tradicional maritime search with 2 agents, and A^* search in two different scenarios considering an environment with two agents;
- tradicional maritime search with 4 agents, and A^* search in two different scenarios considering an environment with four agents;

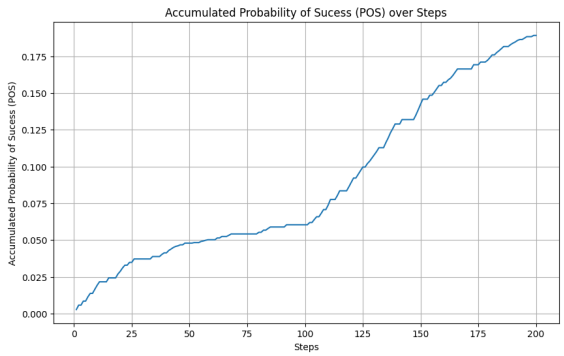
Basically, each group will implement 5 different agents:

1. tradicional maritime search agent (figure 2a);
2. expanding square search agent (figure 3b);
3. traditional maritime search with 2 agents (figure 2b);
4. traditional maritime search with 4 agents (figure 3a), and;
5. informed agent using A^* algorithm.

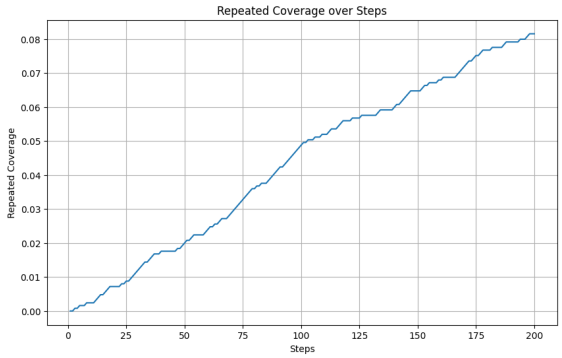
To compare the results, each report must use the following indicators at each step:

- $repeated\_coverage$: the percentage of the grid that has been covered more than once, indicating overlap in search areas, and;
- $accumulated\_pos$: the accumulated probability of success (POS) of the SAR mission; this serves as a way to quantify the chance of finding all SAR targets within a mission.

These data will be used to generate two plots: (i) a comparison of the accumulated probability of success (POS) over steps (figure 4a), and (ii) a comparison of repeated coverage over steps (figure 4b). The first plot illustrates how effectively the algorithm covers the most critical regions, while the second demonstrates the algorithm's efficiency in minimizing unnecessary repetitions. Ideally, the best solution will exhibit a higher accumulated POS and lower repeated coverage.



(a) Accumulated POS over Steps considering an environment with one random walk agent



(b) Accumulates repeated coverage considering an environment with one random walk agent

Figure 4: Results of a random walk agent acting in the configuration number 1

5 Additional material

Listing 2 shows an example of a random walk agent operating under configuration 1. The agent has a time limit of 200 steps [3]. The agent's initial position is set to $(20, 20)$. The data from the agent's steps were stored in a CSV file, which was then used to generate plots 4a and 4b.

```
from DSSE import CoverageDroneSwarmSearch
import pandas as pd

env = CoverageDroneSwarmSearch(
    drone_amount=1,
    render_mode="human",
    prob_matrix_path='../data/config_01.npy',
    timestep_limit=200
)

opt = {
    "drones_positions": [(20, 20)],
}

def random_policy(obs, agents):
    actions = {}
    for agent in agents:
        actions[agent] = env.action_space(agent).sample()
    return actions

observations, info = env.reset(options=opt)

step = 0
```

[3] This time limit applies to all agents in this exercise

```
infos_list = []

while env.agents:
    step += 1
    actions = random_policy(observations, env.agents)
    observations, rewards, terminations, truncations, infos = env.step(actions)
    info = infos['drone0']
    info['step'] = step
    infos_list.append(info)
    print(info)

df = pd.DataFrame(infos_list)
df.to_csv('../results/data_drone_1_config_1.csv', index=False)
```

Listing 2: Simulation example with data collection

References

Dagestad, K.-F., Röhrs, J., Breivik, Ø., and Ådlandsvik, B. (2018). Opendrift v1.0: a generic framework for trajectory modelling. *Geoscientific Model Development*, 11(4):1405–1420.

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.

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.