Final Report - Group 30

Clean Waters

Bernardo Quinteiro
AASMA
Lisbon, Portugal
bernardo.quinteiro@tecnico.ulisboa.pt

Guilherme Saraiva
AASMA
Lisbon, Portugal
guilherme.a.saraiva@tecnico.ulisboa.pt

Sara Ferreira
AASMA
Lisbon, Portugal
sara.c.ferreira@tecnico.ulisboa.pt

ABSTRACT

In this project, we implemented a group of intelligent drones of two different types whose goal is to detect and clean up oil spills in the ocean. One type of drone focuses on scanning the environment, while the other cleans the polluted tiles suggested by the scanner drones as well as scanning its adjacent tiles. The main problem lay in cleaning the polluted tiles as efficiently and as quickly as possible using a multi-agent system while dealing with the spread of said pollution. We solved the problem by using different kinds of reactive architectures, with which we came to a conclusion regarding which one suited this problem best.

1 INTRODUCTION

As ocean pollution becomes a more and more endangering threat to life on Earth, a team of scientists came together to develop a project that would try to minimize the negative effects of this activity. Cleaning the ocean of pollution brings a complex challenge, by employing several variables that will cooperate towards a common goal: fully cleaning the ocean of pollution.

Our solution consists of a simulation of an ocean environment, in which there are oil spills that appear over time. As the oil propagates in the ocean squares, there are two types of agents: **Scanners** (that move through air to find polluted areas) and **Cleaners** (that move through water and will clean the polluted ocean squares). These agents communicate between each other, in order to get a maximized understanding of the scenario and to arrange which drones clean which squares, in order to maximize the amount of squares cleaned and minimize the amount of trips and distance covered. If the oil occupies half of the ocean squares or there are no drones left (due to battery discharge), the simulation ends. The main objective of the project is to develop a **Multi-agent System** where agents have to cooperate to keep as many ocean squares as possible clean.

2 ENVIRONMENT

The environment is a board with an area of 32x32 squares, corresponding to a top view of the ocean area that is taken into consideration for this problem and to which the drones' activity will be restricted. It contains five types of squares: **Scanner Drones** that are represented by a dark blue square; **Cleaner Drones** that are represented by a yellow square; **Charging Stations** that are represented by dark red 3x3 squares; **Ocean squares** that are represented by light-blue squares; **Oil squares** that are represented by black squares. A grid was drawn on the board to aid in the visualization of the zones where agents move.

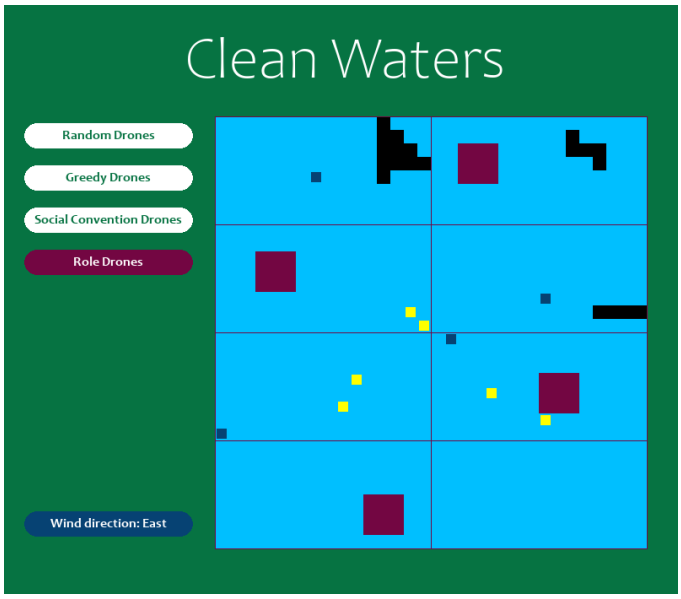

Figure 1: Environment

2.1 Oil

In this environment, there are **oil spills** that spawn in random positions during the whole simulation, ensuring that there will be a **fixed amount of spills per time-step**. These oil spills expand according to the **wind direction and its strength**. Every time-step, each tile has a probability of 10% to expand in a random direction inside the board, whose odds of being selected are influenced by the wind characteristics.

2.2 Properties

The environment is **inaccessible** since the agents don't have complete, accurate and up-to-date information about the whole environment. It is also **Non-deterministic** since the next state of the environment isn't uniquely determined by the current state of the agents and the actions selected by them, as the oil spills expand in a partially random manner. It is **dynamic** as the sea changes as the agents act on it. The environment is also **discrete** because

AASMA Project, IST, 2022, Lisbon, Portugal

https://doi.org/

there's a finite amount of perceptions and actions the agents will sense and make. Finally, it is **episodic** because we divide the world into a series of episodes independent from each other, defining the beginning and end of each run.

3 AGENTS

We have two types of agents, each with their own internal model with information about the environment and each agent's state. Its attributes and how they'll support these agents' approach on solving this problem will be further described in this section.

3.1 Scanners

Scanners role is to **detect oil spills that appear throughout the ocean board**. We have **four scanners**, being that each scanner is assigned to 2 of the 8 sections of the board and moves randomly throughout that zone. This type of drone as the following features:

3.1.1 *Sensor Features.*

- Have a big field of view (10) to sense oil spills in the sea
- Detect the agent's amount of battery

3.1.2 *Actuator Features.*

- Send alerts to Cleaners
- Move in a random manner in its zone
- Recharge battery

3.2 Cleaners

Cleaners role is to **clean the oil spills that were detected by scanners**. We have **six cleaners** that behave differently depending on the chosen policy. Furthermore, we chose to add a **small field of view** to this type of agent so that when he is cleaning an oil he senses that there are other oils in that oil spill nearby and cleans them reactively. This type of drone has the following features:

3.2.1 *Sensor.*

- Have a small field of view (3) to sense oil spills in the sea
- Receive alerts from sensors
- Detect the agent's amount of battery

3.2.2 *Actuator.*

- Move in the direction of the oil spills
- Clean oil spills
- Charge battery

4 SYSTEM ARCHITECTURES

The following UML represents the overall **architecture** and **class relationships** presented on our solution:

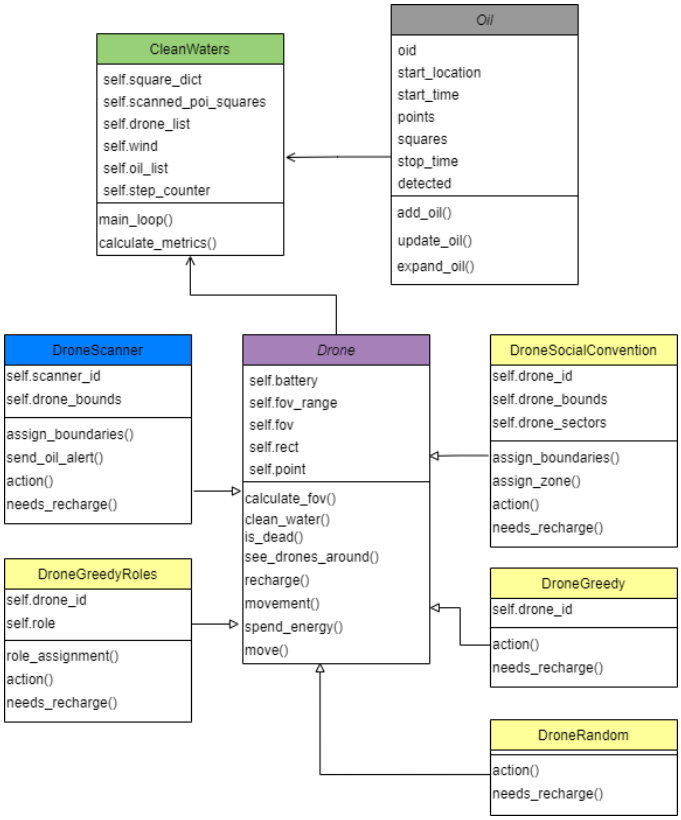

Figure 2: UML Architecture

Our solution consists of the main execution block and various classes: a main class **CleanWaters** containing the main loop and board initialization; a superclass **Drone** that has variables and methods common for all different implementations of Cleaner agents (DroneRandom, DroneGreedy, DroneSocialConvention and DroneRoles) and classes that constitute the environment, specially the **Oil** class as it greatly influences our problem.

4.1 Scanner Reactive Agents

The implementation of the Scanner drones will be the same in the other architectures except for the Random Agents, as they only clean up the oil spill when it crosses through them. These agents, besides moving randomly, have a small Social Convention component since they can only move in assigned zones to maximize the observation of the board. Besides that, when they detect a new oil spill, they broadcast that information to the Cleaners. It's important to note that they are able to detect oil in zones that are not assigned to them.

```
FOR point IN fov:
    IF point.with_oil or point.is_recharger:
        ADD point TO observed_points_of_interest
        SEND oil alert

    ELSE IF current_point == point.is_recharger \
```

```
            AND needs_recharge():
        Recharge()

    ELSE:
        MOVE randomly in assigned zones
```

4.2 Random Agents

This is the simplest architecture, its only purposes were providing insights when doing the comparative analysis in section 6 as well as building a base to develop the other more complex architectures. Here we do not have scanners but have the six cleaner agents who move **randomly** throughout the environment and clean oil squares if they bump into them and recharge themselves if their battery is low, without any purpose.

```
IF current_point == point.with_oil:
    Clean()

ELSE IF current_point == point.is_recharger \
        AND needs_recharge():
    Recharge()

ELSE:
    MOVE randomly
```

4.3 Reactive Agents

On top of the Random Agents architecture, we built the Reactive Cleaner agents using a Greedy policy and the final version of the Scanner Agents. Each of the four Scanners has an id that determines the zone where they are allowed to move, and they move randomly. When they detect that a square has oil, that one is added to a dictionary that contains all squares that have oil that were detected. The Cleaner agents act in the following manner:

```
IF current_point == point.with_oil:
    Clean()

ELSE IF current_point == point.is_recharger \
        AND needs_recharge():
    Recharge()

ELSE:
    FOR point IN observed_points:
        IF point.is_recharger AND needs_recharge():
            ADD point TO POIs[0]
        ELSE IF point.with_oil:
            ADD point TO POIs[1]

    IF POIs[0]:
        GET directions TO closest_point(POIs[0])
        ADD directions TO directions_list
    IF POIs[1]:
        GET directions TO closest_point(POIs[1])
        ADD directions TO directions_list

    FOR directions IN directions_list:
        IF directions NOT IN drones_around():
            MOVE to random direction IN directions
            RETURN
```

With this implementation, we are prioritizing the drone's **battery** in comparison to the oil spill emergence as we put the recharger points in first place in the Points of Interest (POIs) list, this choice enable Cleaner Drones to last longer.

4.4 Reactive with Social Convention Agents

This architecture is similar to the Reactive Agents with the difference that these drones are **assigned to 2 zones** of the environment. We added this as we believed it would greatly improve the amount of steps taken to reach each oil spill, as each one would have at least one agent close to him and the cleaner would be able to fully clean it almost immediately. The distribution of zones are similar to the ones assigned to the Scanners; the remaining two agents were assigned to the four zones in the middle of the board when following its y axis, one cleaning the left side and the other the right side.

4.5 Reactive with Roles Agents

In this architecture, we assigned **roles** to the drone agents as an alternative to using social conventions as a coordination mechanism. When an agent is assigned to a role in a specific state, some of its actions may become unavailable. Therefore, **every time-step**, a role is assigned to all drones based on a **potential function** that corresponds to the distance between the drone position and the closest point of each oil spill. Thus, the respective drone's role will be the closest oil spill in relation to other drones. Besides that, the drones **split** into teams with the same role in order to maximize the cleaning of the oil. It's worth noting that each drone is aware of its potential function and is capable of assigning a role to each drone. As a result, each of the agents' roles can be assigned in parallel and without communication.

```
IF current_point == point.with_oil:
    Clean()

ELSE IF current_point == point.is_recharger \
        AND needs_recharge():
    Recharge()

ELSE:
    role = role_assignment()

    FOR point IN observed_points:
        IF point.is_recharger AND needs_recharge():
            ADD point TO POIs[0]
       ELSE IF point IN role.points AND point.with_oil:
            ADD point TO POIs[1]

        IF POIs[0]:
            GET directions TO closest_point(POIs[0]))
            ADD directions TO directions_list

        IF POIs[1]:
            GET directions TO closest_point(POIs[1]))
```

```
        ADD directions TO directions_list

    FOR directions IN directions_list:
        IF directions NOT IN drones_around():
            MOVE to random direction IN directions
            RETURN
```

The only difference between this implementation and the Simple Reactive one is that we call the role (oil spill) assignment before the movement per se and then, instead of just moving to the closest point of interest, when it has sufficient battery, it moves towards the correspondent closest point of the assigned oil spill.

5 EXPECTATIONS

Entering the testing phase, we expect the **Greedy w/ Roles** architecture to decisively out-perform the other architectures, with the Greedy w/Social Convention being the second most effective, followed by the Greedy architecture and, finally, the random architecture.
This prediction comes from the expectation that higher-level coordination will be more successful: Random drones have no coordination whatsoever; Greedy drones have no coordination with each other, as they only go after the nearest known oil square; Greedy w/ Social Convention drones are coordinated with each other, although in a somewhat simple way: each one does its task and its task only, with no possible adaptation due to circumstances and, finally; Greedy w/ Roles drones have total coordination, as they are constantly trying to figure out what is the optimal way of approaching the current known scenario: how many drones to attribute to each oil spill and which ones.

6 COMPARATIVE ANALYSIS

To test and evaluate each architecture's performance, we defined the following set of metrics (averages):

- **Percentage of Ocean Squares per Time-Step**;
- **Total Number of Cleaned Squares**;
- **Percentage of Oil Squares in the final step**;
- **Average duration of oil spills**;
- **Total number of oil spills created**;
- **Total time steps until end of simulation**;

First, on section 6.1, we tested with the previously defined amount of active oil spills at each time step of 3. Then, on section 6.2 tests were performed with 5 oil spills.

6.1 Experiments with 3 oil spills at the same time

We performed a total of **30** runs for each architecture, to which we obtained the following results:

6.1.1 Percentage of Ocean Squares per Time-Step

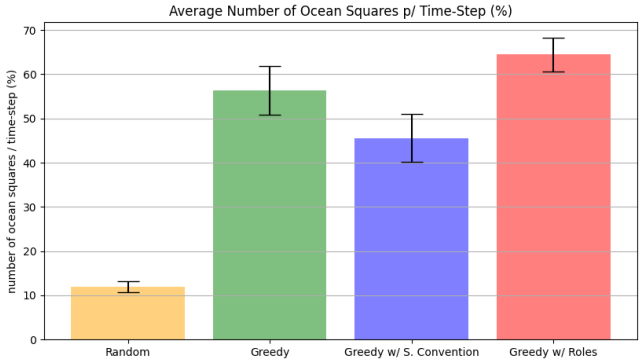

Figure 3: Percentage of Ocean Squares per Time-Step

As the main objective of this scenario was to keep the oceans as clean as possible, this metric is the most important one to assess which architecture is the most effective. From the results displayed above, we can see that the drones that better suited the problem were, from best to worst, Greedy w/ Roles, Greedy, Greedy w/S. Convention and Random. Now, in order to assess the reason behind this, we can observe the results of these tests when applied to our secondary metrics, which analyze more specific aspects of the behaviors that come from each architecture;

6.1.2 Total Number of Cleaned Squares

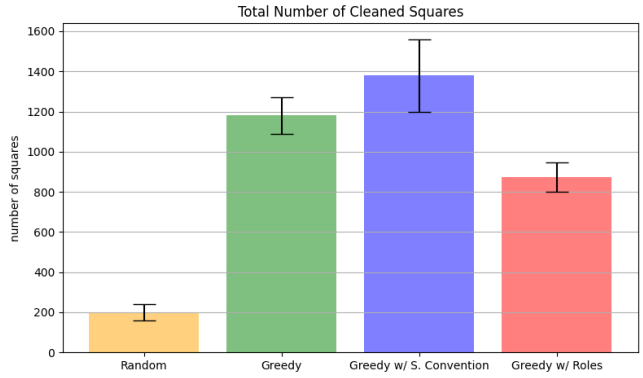

Figure 4: Total Number of Cleaned Squares

In this metric, somewhat contradicting the previous one, the Greedy Drones w/ Social Convention were the ones that cleaned the most tiles, in total. This is due to the fact that, given the nature of each drone's only filling certain zones, what will happen in several scenarios is that, as an oil spill becomes untameable for a drone, for example, in zone 6, the drones in the surrounding will be very much capable of cleansing the oil squares that occasionally appear in their zone, coming from there. With this, most drones will have a very high tally of cleansed squares, but will not be able to end the spill in its entirety, only contain its spread. On the other hand, greedy drones make much more thorough "attacks" on active oil spills, as there are no spatial restrictions for their movement:

Once they go to an active spill, they'll probably only finish once the spill is fully dealt with, which will lead to fewer occurrences of situations like the ones in Greedy w/ Social Convention, where the drones are cleaning squares without dealing with the underlying issue that is making them constantly appear that is the oil spill in itself, that continuously grows unattended.

6.1.3 Percentage of Oil Squares in the final step

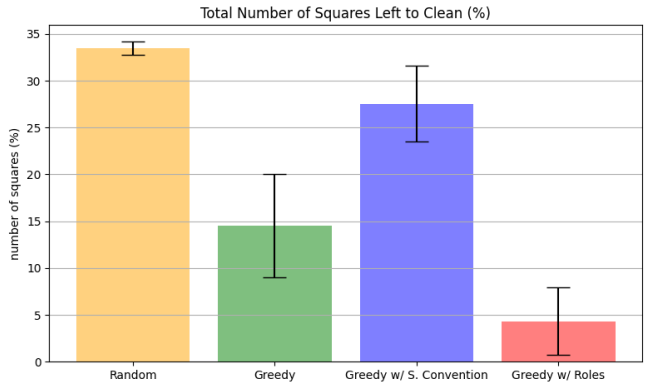

Figure 5: Percentage of Oil Squares in the final step

This metric's results are somewhat confirmatory of the results of the main metric, as they show that random drones, as they usually lose the game, will have many squares left on their last step. Greedy w/ Social Convention drones follow them, for the same reasons we mentioned in the previous metric: how this architecture will lead to some overloaded zones, although, most of the time, not enough to actually end the game. Then, what creates the massive separation between the Greedy and Greedy w/ Roles is the superior organization of the Greedy w/Roles when assigning which drones go to which spills. Furthermore, when analyzing the Greedy w/ Social Convention drones, if a cleaner runs out of battery and dies, there's a chance that no other agent will be in a certain zone that the one that died covered, so it's impossible to clean that area, but oil spills will still be cleaned in other zones and won't be able to expand.

6.1.4 Average duration of oil spills

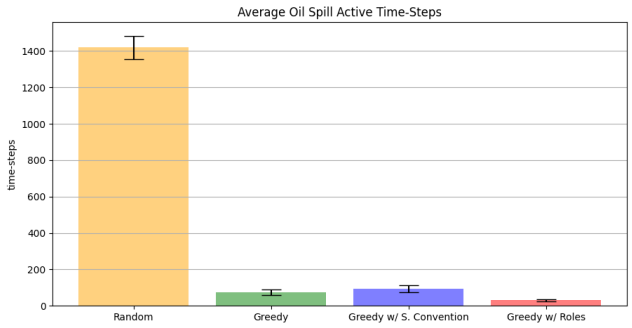

Figure 6: Average duration of oil spills

From these results, the one that stands out, although expectable, is one of the random drones that almost never manage to clean up one entire oil spill. From the other three, the one that struggles the most to fully clean an oil spill is the Greedy w Social Convention drones, given the scenario mentioned in the previous analysis. Then, the Greedy w/ roles clearly is more effective at cleaning oil spills, given its separation of drones per number of active oil spills, in order to terminate them all as effectively as possible.

6.1.5 Total number of oil spills created

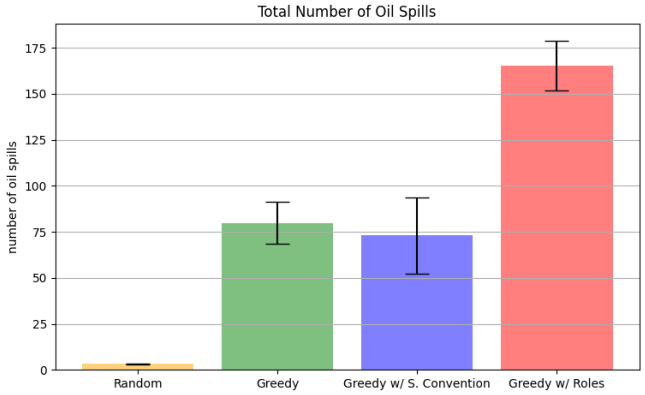

Figure 7: Total number of oil spills created

These results are highly related to the one of the "Average duration of oil spills" results, given the cap of 3 concurrent oil spills on the map at the same time (minimum two). Therefore, there is a reverse proportionality relation between these two statistics that only doubles down on the superior effectiveness of Greedy w/ Roles drones in fully cleansing oil spills, as effective cleansing leads to the appearance of more oil spills. In contrast, due to the random drones' inability to clear oil spills, they only have 3 oil spills created all simulation long.

6.1.6 Total time steps until end of simulation

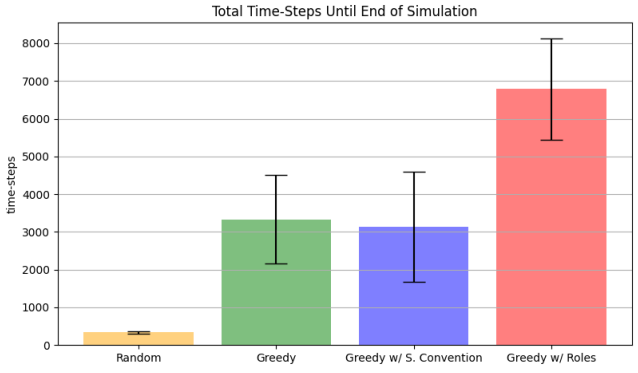

Figure 8: Total time steps until end of simulation

This final metric highlights each architecture's ability to remain "alive" for as long as possible in the game—by having at least one functional drone and not having over half the ocean squares be oil squares. With the results viewed above, we can verify the results seen in the previous metrics and further establish this hierarchy of effectiveness.

6.2 Experiments with 5 oil spills at the same time

For further tests, we decided to keep the same metrics as before but increased to 5 the amount of active oil spills at the same time, since we did not know how our solutions would behave in different scenarios, specially Greedy w/ Roles drones as we expected the differences between them and Greedy w/ Social Convention drones would be less abrupt. In this section we'll discuss the most interesting results we've got:

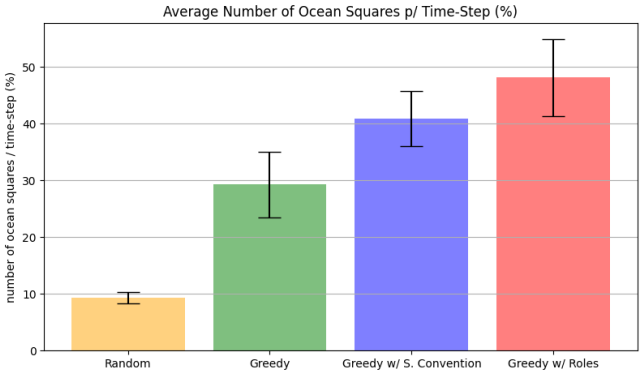

Figure 9: Percentage of Ocean Squares per Time-Step

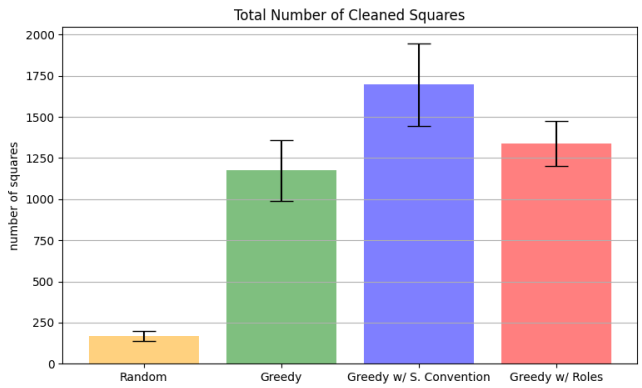

Figure 10: Total Number of Cleaned Squares

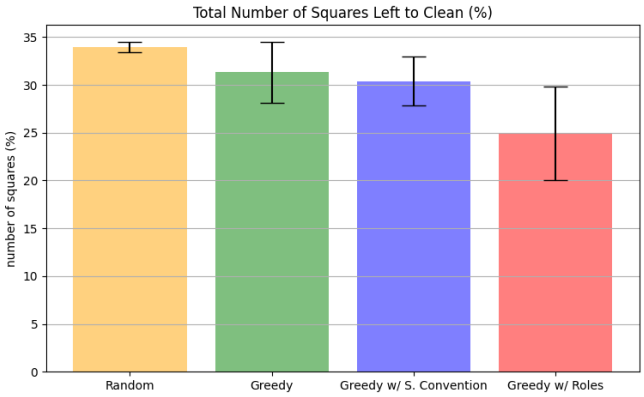

Figure 11: Percentage of Oil Squares in the final step

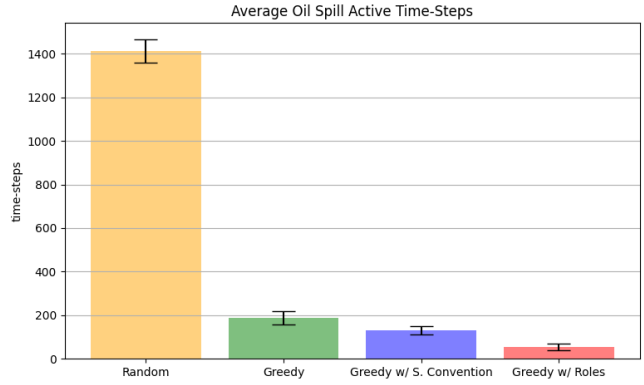

Figure 12: Average duration of oil spills

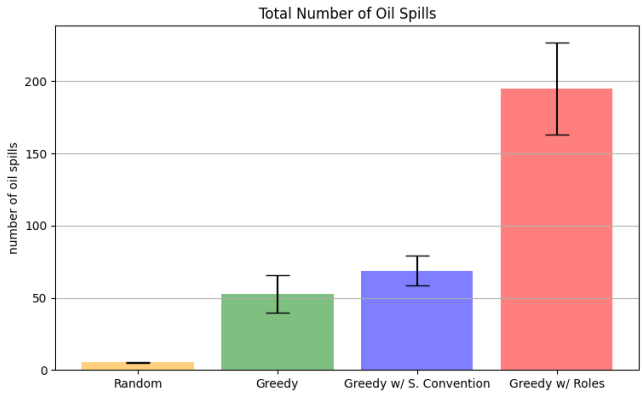

Figure 13: Total number of oil spills created

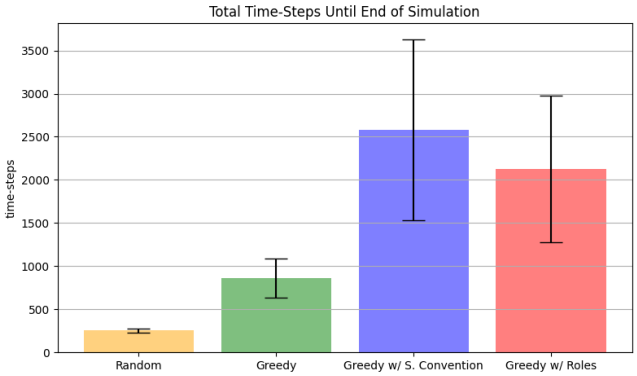

Figure 14: Total time steps until end of simulation

In these results, we observed a much better performance from the Greedy W/ Social Convention drones, compared to the Greedy Drones, even having categories where these appear to perform better than the Greedy w/ Roles drones.
As the number of active oil spills at the same time increases, also does the probability of these being evenly or close to evenly dispersed. Therefore, the organization of the Greedy w/ Social Convention drones will benefit, as there will almost never be drones that have no task assigned and they will all be evenly distributed across the map to deal with these threats, opposed to the greedy drones, that, now, with the amount of active oil spills, will struggle even further to have optimal action, due to its lack of coordination. Regarding the Greedy w/ Roles drones, with five oil spills what happens is that two of them "attack" one oil spill while the other four split themselves and each oil spill is cleaned by one agent. This makes the behaviour of the Greedy w/ Social Convention more similar to the Greedy w/ Roles than with the Greedy.

7 CONCLUSIONS AND FUTURE WORK

From the comparative analysis made above, we came to the conclusion that the difference in effectiveness at completing the simulation's objectives was, somewhat explicit: (from most to least effective) Greedy w/ Roles acted more intelligently so reacted best in most situations and Random Drones were obviously the worst. The comparison between the Greedy and Greedy w/ Social Convention is more complex to define as they behave differently depending the scenario, having their pros and cons. Greedy w/ Social Convention are flexible to an increased number of oil spills, but do not clean oil spills rapidly, unlike Greedy Drones who behave more efficiently with a small amount of oil spills. This hierarchy was verified in most metrics used, except for the one of the most cleaned squares, which is justifiable by the fact that both Greedy and Greedy w Social Convention drones will make many trips and clean more oil, although not in a way that mitigates the oil spills' expansion in optimal fashion, leading to a more broad presence of oil squares.

The main surprise, for us, was that in the experiment with 3 oil spills, Greedy drones out-performed the Greedy W/ Social Convention Drones, as we expected the fact that the latter's organization among themselves would lead to them having a more effective behavior in dealing with this problem. However, in the scenario of 5 active oil spills, the Greedy w/ Social Convention drones were indeed more effective than the Greedy drones, as expected. But we cannot conclude a definitive comparison between them as in most of the metrics we defined the standard errors overlap with each other. We could obtain different results that could probably give us a better idea of which of these architectures is most effective for this problem by using another sample or by adding more runs to the tests. Other than that, the decisively superior results of the Greedy w/ Roles drones and the decisively inferior results of the random drones were fully expected.

As for further work, we'd like to implement more variables that could make this problem a more complex one, like having another type of "negative" square to add to the oil squares, with different behavior, such as plastic or other types of sea pollution. To these new squares we could assign different priorities and incorporate a new utility function like priority of plan choosing, for example.

On top of that, we'd also improve our project by adding more efficient architectures, such as q-learning and hybrid. This would also prove to be a massive upgrade on the current solutions, in our opinion, if we were to add the new variables to the problem, mentioned partially in the paragraph above. The drones could learn how oil spills behave and define strategies with the goal of efficiently surround the spills, "attack" and clean them in a small amount of time-steps.