SaFFi - SwArm FireFighters

A survey on firefighter robots in Argos

https://github.com/chiara-volonnino/irs-19-SaFFi

November 2019

Nicola Atti, nicola.atti@studio.unibo.it, matr: 857726
Chiara Volonnino, chiara.volonnino@studio.unibo.it, matr: 843650

Intelligent Robotic Systems
A.A. 2018-2019

Index

Introduction

With this project we are going to tackle the use of a robot swarm to perform the typical actions needed in order to extinguish a fire, normally taken by human firefighters.

First of all, a robot swarm is a particular case of use robotics, where a high number of robots, usually of small dimensions and limited capabilities, cooperate together in order to achieve a more complex goals.
Typically these swarms are used in high risk contexts, in order to exploit the high number of robots in case of failures and hazards.
Each robot should be able to navigate an unknown environment, while avoiding obstacles and locating fires. The swarm should split equally to efficiently deal with multiple fires present in the environment.

This idea stems from the necessity of aiding the firefighters during risky operations, such as collapsing structures, high presence of smoke or harmful chemicals in the atmosphere.

In order to explore the possible solutions for this problem we used ARGoS as our simulation environment, since it allows us to simulate efficiently large-scale robot swarms.

Chapter 1

Requirements Engineering

In this chapter we are going to define the requirements of the simulation for our case of study.

1.1 Environment Requirements

The environment should be configured in order to:

- ➢ Simulate the presence of one or more fires making them visible to the robots.
- ➢ Simulate the increasing temperature reading while approaching a fire.
- ➢ Have the option of adding walls or obstacles.

1.2 Firefighter Swarm Requirements

Each robot composing the firefighters swarm must show the following behaviours:

- ➢ Locate and move to the fires present in the environment.
- ➢ Avoid collisions with obstacles or other robots.
- ➢ Avoid fires which have already been dealt by a pre-determined number of robots.
- ➢ Stopping near the fires at a safe distance determined by their temperature reading.

Chapter 2

Solutions

In this chapter we are going to illustrate our design process used to find the right solutions for this type of problem, focusing on the issues we encountered and the ways we solved them.

2.1 Premises

Since ARGoS still lacks some simulated sensors and actuators, we had to improvise the ones we needed with the ones that the simulator offered us.
Furthermore ARGoS doesn't permit any run-time modification to the environment once the simulation has started, removing the possibility of designing the simulation of a more realistic case of study, so there is no chance of removing the fires after they have been extinguished by the swarm.

2.2 Solution 0

At first, as solution, we considered a **subsumption architecture** for our robot's controller creating behaviour levels to manage the needed operations; then we realised that we needed to consider each behaviour every time, and also that inhibiting lower behaviour levels would produce unwanted actions.

2.3 Solution 1

After our first idea we decided to use a **fusion architecture**, based on specific motor schemas, to express the movement behaviour of each robot, wrapped into a **finite state automaton (FSA)**, in order to separate the different actions that the robot should take to resolve the given task.

2.3.1 Temperature and fire simulator

A crucial part in designing our solution was creating the environment, more specifically how to simulate the presence of fires and how to make the robots attracted to them.
As mentioned before, ARGoS doesn't offer any temperature source or sensors, so we decided to simulate the presence of a fire [Figure 2.1] by:

- ➢ Changing the arena floor by adding *concentric circles* of different colours, representing different temperature areas, mimicking the temperature variation that a fire normally produces. We used an *orange* colour to represent a zone of high temperature in the fire's

proximity, and a *red* colour to indicate the origin point of the fire, a place where a robot shouldn't enter.

➢ Adding above the centre of those circles a *light*, at maximum intensity, representing the light produced by the flames. The robots will be attracted by those lights.

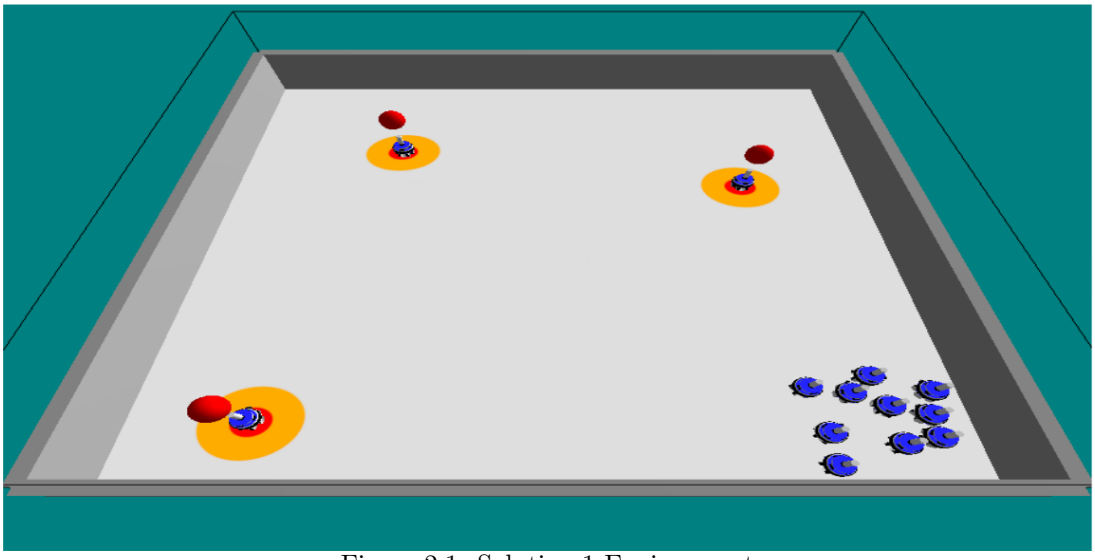

Figure 2.1: Solution 1 Environment

2.3.2 Control Architecture

As shown in figure 2.2 the robots can transit through those different states:

➢ **Search fire**: in this state the robot must deal with finding the different fires in the given environment, while avoiding any obstacle or other robots found on its way.

➢ **Attempt Neighbours (Wait for help)**: the robot transits to this state when it reaches a fire, by entering the high temperature zone. While they're in this state they constantly broadcast their presence at every step and then check if the number of neighbouring robots in their same state has achieved the number of robots needed to begin extinguishing the fire.

➢ **Deal fire (Extinguish Fire)**: the robot transits to this state when the number of robots near a fire achieves the number needed to begin extinguish it. When a robot reaches this state it begins to push away any other robot that's approaching its position, as they are not needed anymore and should move to a different area of the environment.

2.3.3 State 0: Search Fire

This is the initial state of each robot when the simulation begins. In this state the only task they have is to find the nearest fire and move towards it, avoiding every obstacle and moving away from any fire that has achieved the necessary number of robots.

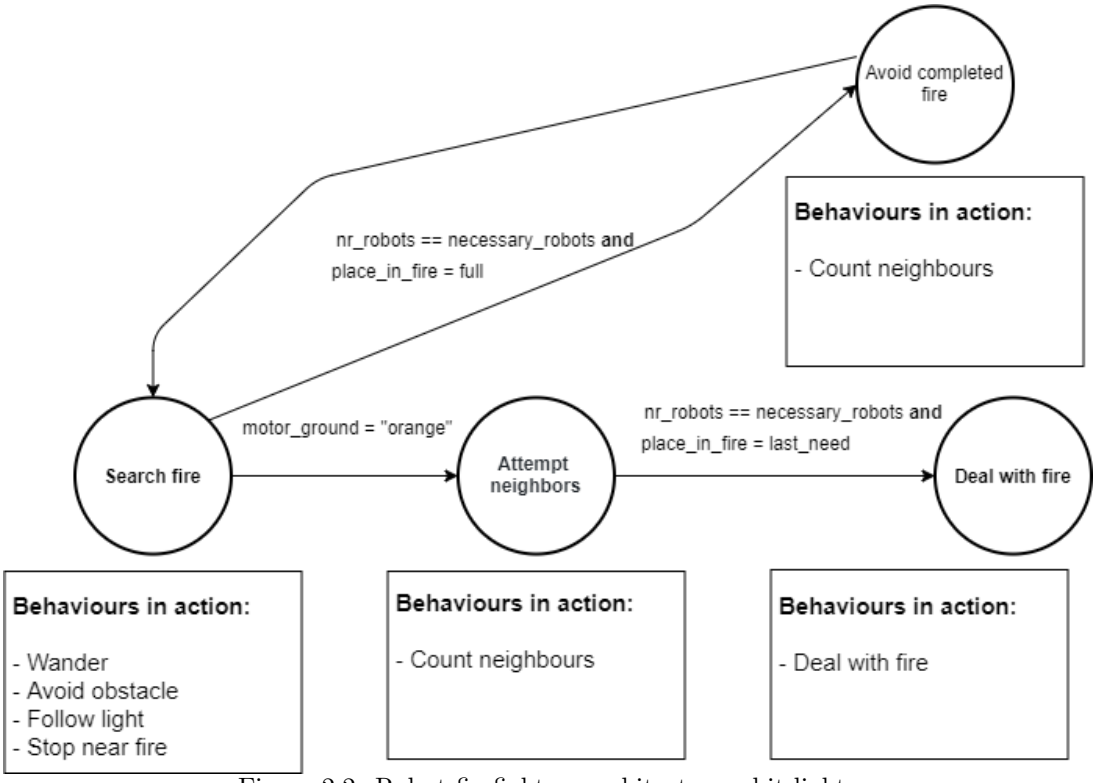

Figure 2.2: Robot firefighters architecture whit lights

This has been achieved using a fusion architecture, with a motor schema composed by the union of the vectors generated by the potential fields that we are going to describe.

2.3.4 Motor Schema

Wander

This is the basic force that always moves the robots in a random direction, in order to pick up the light signal.

Follow light

The robots are directed towards a light, through a vector with the higher intensity the further they are from the light (decreasing the closer they get) and with a direction corresponding to the angle of the sensor with the highest intensity.

Avoid obstacle

The robot is directed away from an upcoming obstacle, based on a vector with opposite direction of the proximity sensor with the highest reading.
After some experimentation we also added a tangential vector (perpendicular to the repulsive force) in order to guide the robot around an obstacle.

Avoid complete fire

This vector is present if the robot is approaching a fire that is being dealt by other robots. The robot is directed away by a vector opposite to the closest robot in state two, obtained by communicating with the range_and_bearing sensor. This vector has a fixed intensity and persists until the robot leaves a given range.

2.3.5 Robot Operations

Stop near fire

When approaching a fire each robot is constantly checking the value of his `motor_ground` sensors and, as all sensors read a value that corresponds to the colour orange (in our solution is 0.6949372534658) it stops and transits to the next state, as it has reached the high temperature zone of a non-completed fire. After some tests we found out that a strict equality constraint was too tight and caused that the robots would stop in the wrong places, to solve this problem we checked if the `motor_ground` readings were inside a range of values (between 0.2 and 0.7).

2.3.6 State 1: Attempt Neighbours (Wait for help)

When a robot is in this state it means it has reached a fire and the number of robots needed has not been reached yet.
In this state the robot remains completely still, and it broadcasts at every step, through range and bearing its presence to its neighbours, until the number of robots near a fire reaches the critical number, then each one transits to state 2.

Observation

A robot cannot obtain data from another through `range_and_bearing` if it is not in line of sight with the broadcaster. Sometimes, depending on the position of the robots around a fire, it happens that some of them can't see each other preventing them from counting the number of neighbours correctly. As we are using this sensor to emulate some form of radio communication we had to come up with a solution to prevent the loss of sight.
We added another robot, named **robot_antenna**, directly under the light in the centre of the fire, with the single task of gathering the `range_and_bearing` readings from its neighbouring firefighter robots, and broadcasting to them the command to change state when it counts necessary robots.
This solution works because this robot can always see its neighbours.

2.3.7 State 2: Deal with fire

This is the final state that a robot can reach, and will never transits out of it. During the rest of the simulation the robot will always broadcast through `range_and_bearing` that its position the fire has archived the number of needed robots, repelling other near robots that are in state 0.

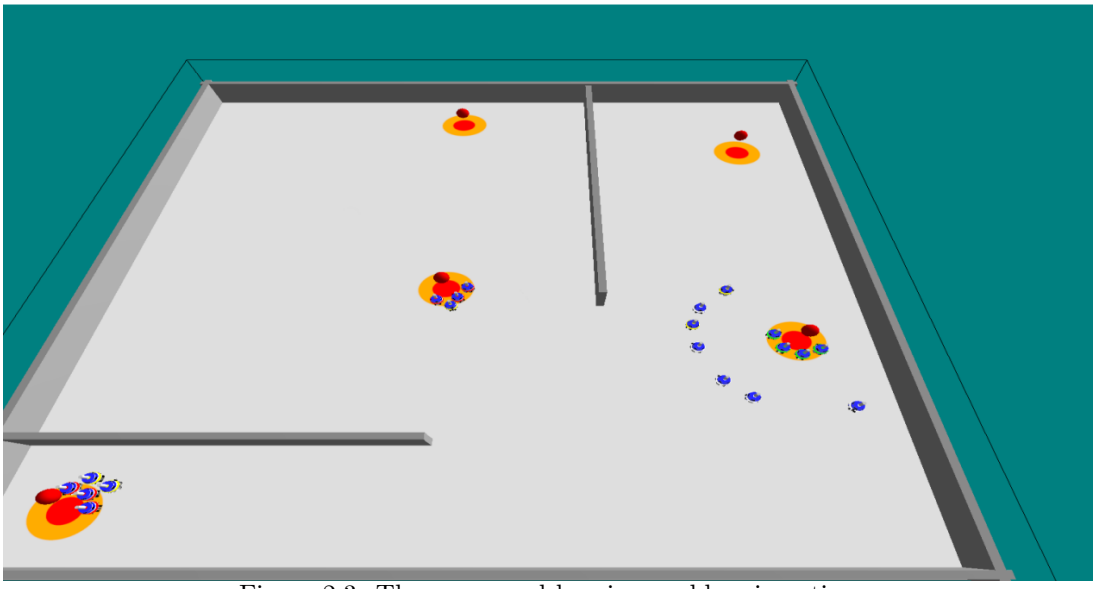

Figure 2.3: The range_and_bearing problem in action

2.3.8 The "conga" problem

During our tests we encountered an anomaly regarding the maximum range of reception of the `range_and_bearing` sensor, which is perceivable by the other robots only when within circa 100 cm, although they could communicate with the antenna robot from a lot further away.
This caused the robots to orbit around the completed fire [Figure 2.3] and to never get too close to another light in order to perceive it as the most intense.

Explored Solutions

At first we tried to find a way to manually increase the `range_and_bearing` range, but with no success as ARGoS doesn't permit it.
Then we tried to increase the strength of the repulsive force, but then the robots began to bump into each other. We also though of other solutions, such as ignoring the light sensors which are on the same side of a completed fire, for a fixed number of steps, but this could lead to unwanted behaviours without solving the problem.
After this tentative we decided that we needed to re-design our solution.

2.3.9 Conclusion

This first solution has been set aside, but it can work if a solution to the problem with the range_and_bearing is found.
We also would like to point out that the swarm can sometimes complete the tasks, but only if a sufficiently large number of robots is given, but this result is only obtained in a non-deterministic way.

2.4 Solution 2

2.4.1 Architecture

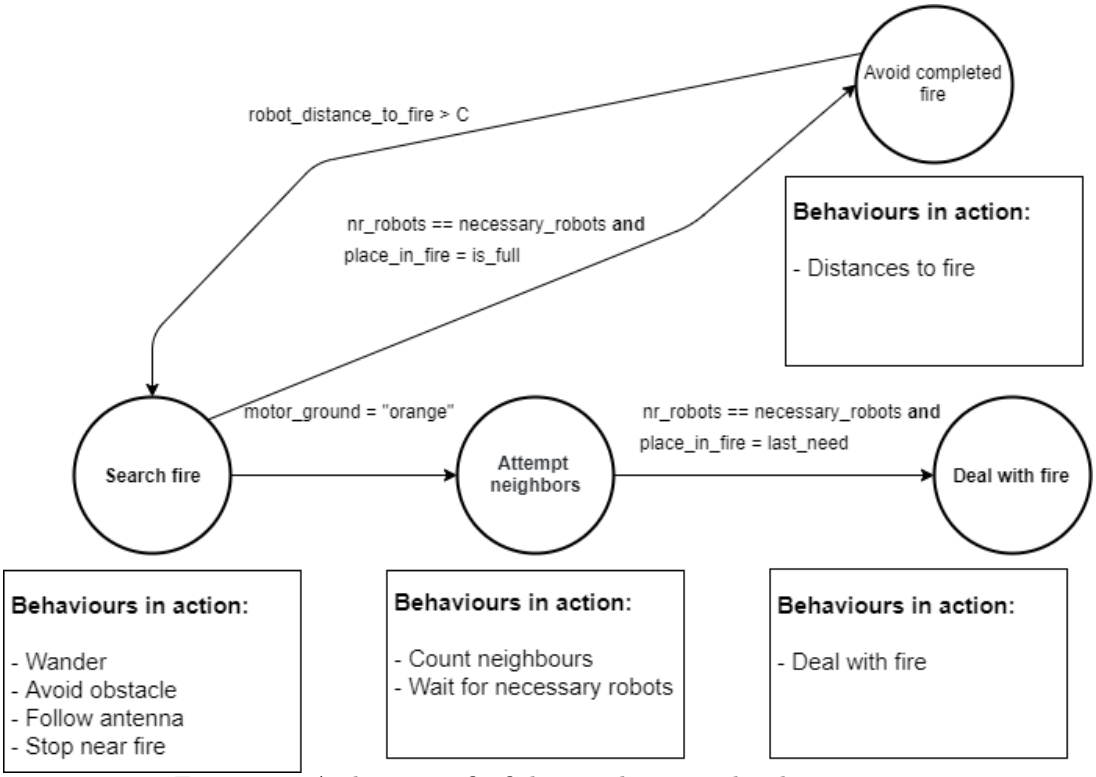

Figure 2.4: Architecture firefighters solution with robot antenna

The architecture of our control system calls back to the previous solution (section 2.3), but with some substantial differences [Figure 2.4]:

- ➢ The lights have been removed from the environment, along with the corresponding motor schema for the robot swarm.
- ➢ The attractive behaviour is now handled by the **antenna robot** (to emulate a long range temperature sensor), with a new motor schema for the firefighter robots.

2.4.2 Robot Antenna

The robot antenna is used to solve the problem we encountered in the previous solution.
Its controller uses a finite state automaton with two states:

- ➢ **State 0**: in this state the antenna broadcasts with its range_and_bearing sensor the presence of a fire, attracting the firefighter robots. While in this state the antenna counts the robots who have reached the fire, and when it reaches the needed number, changes state along with those firefighter robots.

➢ **State 1**: in this state the antenna switches to repelling the firefighter robots away from the completed fire. This solves the problem of the previous solution, since the antenna has a longer broadcast range.

A detailed schema of communications between the different robots in the environment can be found in figure 2.5.

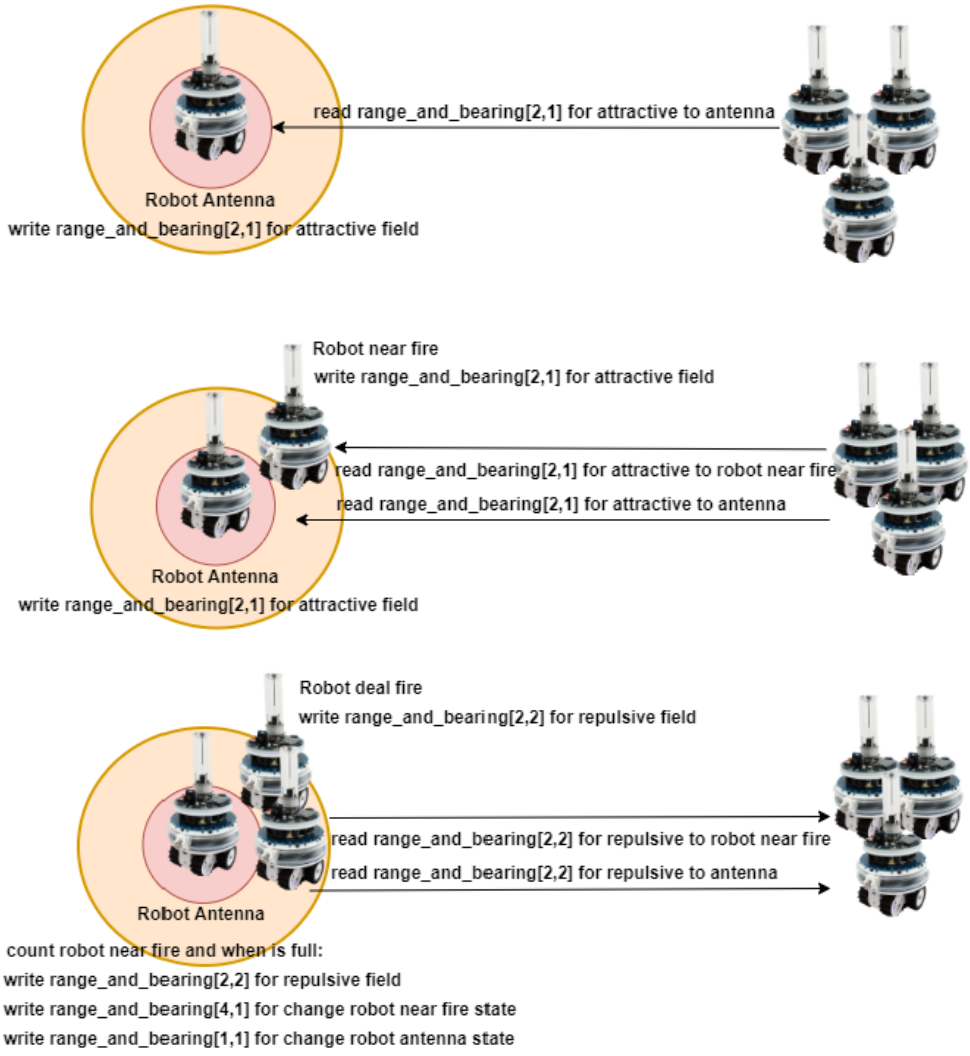

Figure 2.5: Robot communication schema

2.4.3 State 0: Find Fire

This was the only modified state, in particular its motor schema, due to the removal of the lights and the greater use of the range and bearing sensor. The new environment can be found in figure 2.6.

2.4.4 Motor Schema

Follow antenna

The firefighter robots are attracted by the antenna robot, through a vector with the same angle of the closest `range_and_bearing` reading of an antenna attractive field. This vector increases its intensity the further the robot is from the antenna and slows dramatically the closer they get.

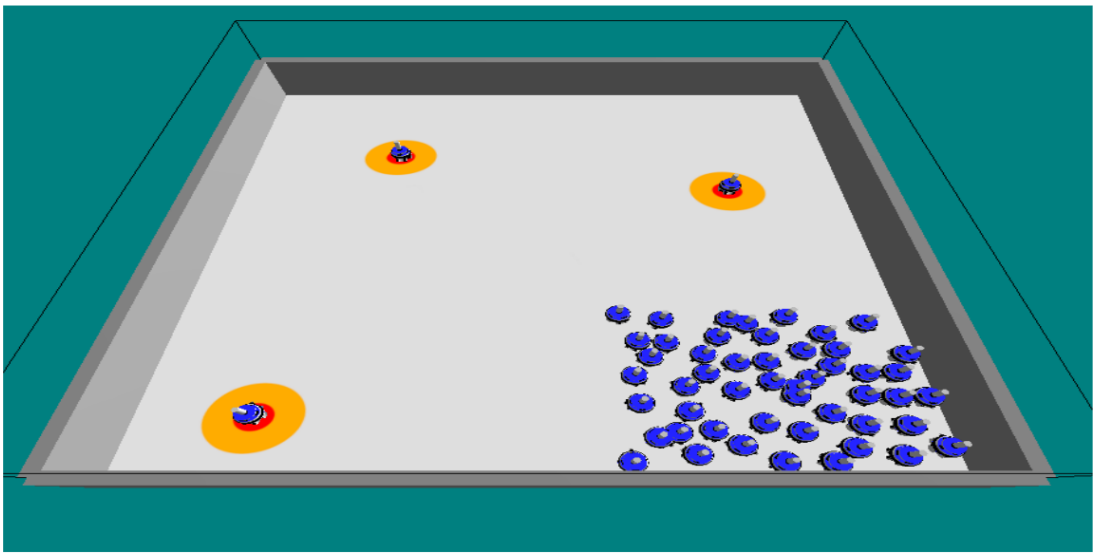

Figure 2.6: Solution 2 Environment

2.5 Solution refinement

After we made sure that the simulation could be completed we began to improve the robot's performances:

- ➢ We added a control on the angle of the proximity sensor for the obstacle avoidance, in order to guide the robot around an obstacle depending on the side they approach. If the robot finds an obstacle to its right the tangential vector will point to his left, otherwise if it finds it on the left it will point to the right, by adding or subtracting 90°.
- ➢ As the robots begin positioning themselves around a fire they will block the line of sight from other wandering robots and the antenna robot, causing them to no longer read its range and bearing broadcast. To alleviate this problem we added the same attractive/repulsive field, written on a different field of the `range_and_bearing` data table, depending on the robot's state. This still slowed the robots due to low range of their range and bearing transmission.

Chapter 3

Performance analysis and test

3.1 Performance

To test the performance of our solution, we executed multiple simulations varying the following parameters:

- ➢ The number of robot composing the swarm.
- ➢ Adding walls to the environment.
- ➢ Adding noise to actuators commands and sensor readings.

The swarm always started from the lower right corner of the arena, trying to simulate a realistic deployment. Each case study was executed at least *10 times*, in order to produce an estimated average performance value.

3.2 Case Study 1 - 50 Robots

We executed the following tests on our 5x5 arena.

Basic

As show in table 3.1 we have that the swarm takes on average 1102 steps before completing the task of finding all the fires and reaching the necessary number of robots on each of them.
This case study will be used as the baseline when assessing the performance of the other test cases.

Walls

Even with this few attempts we observed that the addition of walls, carefully placed to block a robot's direct path from one fire from another, significantly increases the number of steps needed to reach the completion of the task [Table 3.2], with an average of 1469 steps.
We predicted this would happen, mostly because the robots now have to skirt around the walls to reach their destination, but also because the lack of line of sight completely blocks the range_and_bearing sensor (being an infrared communication system), making the robots wander aimlessly until they can see another fire or the other robots.

Noise

In this case of study we added the presence of noise [Table 3.3] on the actuators and sensors of the robot used by our solution, trying to replicate a more realistic scenario, more specifically we added:

- A noise level of 0.1 to the `differential_steering` actuator.
- A noise level of 0.1 to the `motor_ground` sensor.
- A noise level of 0.3 to the `range_and_bearing` sensor.

The noise on the "motor_ground" sensor caused the most amount of problems, causing the robots to sometimes pass over the critical temperature area of a fire, instead of stopping and change their state.
Combined with some packet loss, generated by the noise level on the "range_and_bearing" sensor, this increased the amount of steps needed, with an average of 1499 steps, but we are pleased to say that this didn't affect the swarm's behaviour in a severe way.

Test Nr.	Number of Steps
1	1102
2	821
3	1077
4	971
5	1343
6	1131
7	999
8	1107
9	1322
10	1153

Table 3.1: Results with 50 robots, without walls and without noise

Test Nr.	Number of Steps
1	980
2	2041
3	1544
4	1413
5	1550
6	1296
7	1611
8	1728
9	1473
10	1059

Table 3.2: Results with 50 robots, with walls and without noise

Test Nr.	Number of Steps
1	2016
2	1974
3	1310
4	1100
5	1442
6	1826
7	1520
8	1299
9	1280
10	1221

Table 3.3: Results with 50 robots, with noise

3.3 Case Study 2 - 100 Robots

We executed this test on a 10x10 arena, to spawn the robot swarm from a corner (like the previous series of tests). The number of steps will be significantly higher due to the time needed for the robots to reach the fires.

Basic

By using 100 robots in an environment without noise or walls between the fires we can see that the swarm can complete the task with an average of 4498 steps. Each individual result is shown in table 3.4.

Walls

As shown in table 3.5 by adding walls to the environment the swarm takes an increased number of steps to complete the task, with an average of 4754 steps.

Noise

By adding noise to the simulation, with the same values as the previous tests, the swarm presented the same problems observed previously, and it increased the number of steps needed as expected, shown in table 3.6. In order to complete the task the swarm needed an average of 4907 steps.

Test Nr.	Number of Steps
1	3909
2	5238
3	3774
4	5177
5	4119
6	4042
7	4493
8	3730
9	5242
10	5264

Table 3.4: Results with 100 robots, without walls and without noise

Test Nr.	Number of Steps
1	4858
2	4039
3	5103
4	6192
5	3746
6	3955
7	5563
8	4881
9	4902
10	4303

Table 3.5: Results with 100 robots, with walls and without noise

Test Nr.	Number of Steps
1	4589
2	5384
3	4730
4	4683
5	6307
6	4100
7	4814
8	5700
9	4192
10	4573

Table 3.6: Results with 100 robots, with noise

Chapter 4

Retrospective

4.1 Limits

ARGoS still has some limitations. The inability to modify the environment during run-time and the lack of some sensors forced us to come up with non-optimal ways to emulate what we were missing.
The second solution we found is working as intend and performing well, but the swarm doesn't behave as well when approaching a fire as it was find out in the first solution.

4.2 Conclusion

After performing our tests we can say that increasing the number of robots in our swarm improves the performances in achieving the task.

Overall we are pleased with what we achieved.

Chapter 5

Further Developments

- ➢ Add a system to deal with concurrency problems, such as two robots entering at the same time into the high temperature zone;
  Eg. by making the robot with the lowest identifying number go away and return to the wandering state.
- ➢ Testing another solution using a probabilistic finite state automaton.
- ➢ Refine the calibration of the motor schema vectors, in order to increase the robot's precision and robustness.
- ➢ Add a form of preemptive communication between robots to aid each robot that still hasn't found any fire to move in the right direction sooner, rather than relying on random wander behaviour.
- ➢ Modulate the repulsive behaviour of the robots that are dealing with a fire using their density, increasing the repulsive force perceived by the wandering robots the more robots are perceived in the vicinity.
- ➢ Expand our working solution using advanced techniques, such as mechanisms of reinforcement learning to train the swarm.