Final Project of INFO-H-414 Swarm Intelligence

First Session

28 March, 2024

1 Predator-prey

In this project, we consider a predator-prey scenario, in which a swarm of robots (the *predators*) must capture an intruder (the *prey*). To do so, the swarm must locate the intruder and trap it so that it can no longer move. This requires careful cooperation between the robots in the swarm, as they must coordinate to explore the environment, locate the intruder, and implement a strategy to stop it from moving, while only relying on *local* perception of the environment.

You are asked to produce and assess control software for a group of predator robots. The goal is to maximise the performance of the swarm by designing a cooperative collective behaviour.

1.1 Problem definition

The robot swarm operates in a dodecagonal arena with white floor, as shown in Figure 1. The swarm of predators is initially scattered around the centre of the arena, while the prey is placed in a random position around the predators. The prey will always execute a random walk; that is, it will move in a straight line until it detects an obstacle, at which point it will turn to avoid it. To identify itself to the predators, the prey will keep its LEDs turned on to red. The behaviour of the prey is set through the `prey.lua` script, which you are not allowed to modify.

Goal The goal of the predator swarm is to stop the prey from moving. Each experimental run lasts 10 minutes — 6,000 ARGoS time-steps. The overall performance of the swarm is measured by the number of time-steps the prey remains static. The prey will be considered static if its centre remains within a 10×10 cm^2 area for more than 10 seconds — 100 ARGoS time-steps. Note that this means that the prey must remain trapped for

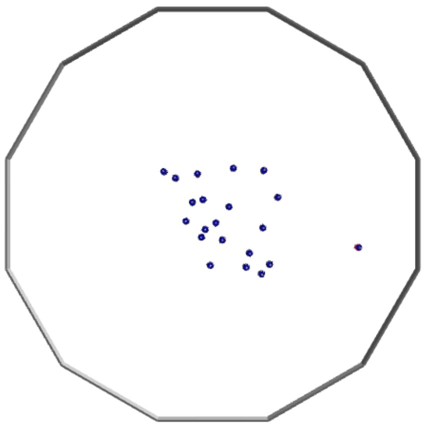

Figure 1: A top view of the arena.

at least 10 seconds for the performance to be greater than zero. More specifically, the performance is computed as

$$\sum_{t=\tau}^{T} X(t)Y(t), \tag{1}$$

where $T = 6,000$ is the total duration of the run, $\tau = 100$ is the minimum number of time steps the prey must remain confined,

$$X(t) = \begin{cases} 1, & \text{if the prey has been confined along the } x\text{-axis} \\ & \text{in the interval } (t-\tau, t] \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

and similarly for $Y(t)$.

ARGoS is configured to automatically dump data on a file whose name can be changed in `predatorprey.argos` (see Section 1.3). This file contains a table with four columns:

- CLOCK: Column indicating the current step.
- TRAPPED: Column indicating if the prey has been trapped for at least 10 seconds (1) or not (0).
- SCORE: Column indicating the current score. Its value is equal to the cumulative sum of the TRAPPED column.
- X: Column indicating the position of the prey along the x-axis.
- Y: Column indicating the position of the prey along the y-axis.

Table 1: Sensors and actuators available to the robots

Available sensors	Available actuators
`robot.proximity`	`robot.wheels`
`robot.range_and_bearing`	`robot.range_and_bearing`
`robot.colored_blob_omnidirectional_camera`	`robot.leds`
`robot.id`	

Swarm composition Originally, the predator swarm consists of 20 homogeneous robots. See Table 1 for a summary of the sensors and actuators available.

Additional Remarks

- The maximal wheel velocity should not exceed 10 cm/s.

In the file `predatorprey.argos`, we provide you with a swarm of 20 robots. You will need to choose the number of predator robots that you want to design your control software for. In order to set your desired number of predator robots, find the lines with the following format (usually line 77):

```
<!-- You can play with the number of predator robots by
changing the 'quantity' attribute -->
```

Change the quantity value on the following line to the desired number of robots. This will place the specified number of robots of that type into the arena.

Additionally, you will also need to modify the range of interaction of the predator robots. To do so, find the lines with the following format:

```
<!-- You can modify their range of interaction by changing
the 'omnidirectional_camera_apperture' and 'rab_range'
attributes -->
```

To keep a constant sensing range for both the omnidirectional camera and the range-and-bearing sensor, first set the value of `rab_range`, and then set the value of `omnidirectional_camera_apperture` to

$$\text{omnidirectional\_camera\_apperture} = \text{arctan2}(\text{rab\_range}, 0.2). \quad (3)$$

1.2 Objective

The goal of this project is to design, implement and evaluate control software that aims to maximise the number of time-steps the prey remains stationary. The control software has to demonstrate a *cooperative* behaviour: one that takes advantage of the swarm's principles. However, this project is not only about optimising the performance of the swarm. Instead, we are also interested in seeing solutions that make use of the swarm robotics principles that you learn during the course. You are also asked to investigate two important properties of robot swarms: scalability and locality. To this end, you will conduct experiments while the robots operate with the control software you designed. Since the performance is stochastic, you will need to run your control software multiple times in each setting to get a reliable estimate of the performance. For the experiments below, run ARGoS 10 times on different seeds. In your report, please report the results obtained from the experiments (the performance) in the form of boxplots, where each box corresponds to the ten repetitions.

In order to investigate the **scalability** of the swarm, you will test the control software that you developed in a varying number of swarm sizes. These experiments must be performed with your original implementation and *without* further modification or adaptation to the new sizes.

Try to answer the following questions when analysing the results: How does the performance scale with the number of robots? Is the performance always increasing? Are there certain sizes for which the performance of the robots varies more notably than in others? What are the possible reasons that would explain the observed development of the performance?

You are asked to provide a comparison of at least ten swarm sizes with which you are able to address the aforementioned questions. Bonus points are given if the studies are conducted with a higher resolution of swarm sizes. Keep in mind that, in order to obtain data that is relevant for analysis, the variations in size should be appropriate. Sequentially incrementing/reducing the number of predator robots by 5-10 is a good rule of thumb. However, remember to switch off the visualisation of ARGoS (see Section 1.3) when you experiment with large swarm sizes. Otherwise, your experiments will take very long.

To investigate the effect of **locality** in your solution, you will follow a similar approach. Given the sensors and actuators specified in Table 1, we can make the predator robots interact more or less locally by modifying the range of the omnidirectional camera and the range-and-bearing sensor. By increasing their range, the robots will be able to communicate with peers that are farther away. Conversely, reducing their range will force the predators to interact only with their nearest neighbours.

For your analysis, try to follow the same reasoning as for the one on scalability. Make sure to follow the approach outlined in Section 1.1 to scale the interaction range appropriately.

Remember to follow the principles of swarm robotics and apply what you learn during the course. Simple and reactive behaviours tend to work well and are very representative of the swarm robotics principles. Be aware that, for the project evaluation, the analysis is as important as the implementation. Make sure that the information provided in the report is meaningful, clearly written and complete. A good analysis is the one that discusses how design choices affect the performance of the robots — which are supported by the results.

1.3 Setting up ARGoS

- Download the experiment files: SR_Project_H414.zip
- Unpack the archive into your $ARGOS_INSTALL_PATH directory and compile the code

  ```
  $ unzip SR_Project_H414.zip         # Unpacking
  $ cd SR_Project_H414                # Enter the directory
  $ ./build.sh                        # Compile the code
  ```

- Set the environment variable ARGOS_PLUGIN_PATH to the full path in which the build/ directory is located:

  ```
  $ export ARGOS_PLUGIN_PATH=\
  $ARGOS_INSTALL_PATH/SR_Project_H414/build/:$ARGOS_PLUGIN_PATH
  ```

  You can also put this line into your $HOME/.bashrc file, so it will be automatically executed every time you open a terminal.

- Run the experiment to check that everything is OK:

  ```
  $ cd $HOME/SR_Project_H414        # Enter the directory
  $ argos3 -c predatorprey.argos    # Run the experiment
  ```

If the usual ARGoS GUI appears, you're ready to go.

Switching the visualization on and off. The experiment configuration file allows you to launch ARGoS both with and without visualization. When you launch ARGoS with the visualization, you can program the robots interactively exactly like you did during the course. Launching ARGoS without the visualization allows you to run multiple repetitions of an experiment automatically, e.g., through a script. By default, the script launches ARGoS in interactive mode. To switch the visualization off, just substitute the visualization section with: <visualization />, or, equivalently, comment out the entire qt-opengl section.

Loading a script at init time. When you launch ARGoS without visualization, you cannot use the GUI to set the running script. However, you can modify the XML configuration file to load automatically a script for you. At line 44 of `predatorprey.argos` you will see that the Lua controller has an empty section <params />. An example of how to set the script is at line 47 of the same file. Just change line 44 to the example and set the script attribute to the file name of your script.

Changing the random seed. When you want to run multiple repetitions of an experiment, it is necessary to change the random seed every time. To change the random seed, set the value at line 11 of `predatorprey.argos`, attribute random_seed.

Making ARGoS run faster. Sometimes ARGoS is a little slow, especially when many robots and many sensors are being simulated. Sometimes you can make ARGoS go faster by setting the attribute threads at line 9 of `predatorprey.argos`. Experiment with the values, because the best setting depends on your computer.

Changing the name of the log file. If you want to set a specific name for the log file, other than `output.csv`, modify the output attribute in the <loop_functions /> tag, in line 17 of `predatorprey.argos`.

2 Deliverables

The deliverable will be a zip file containing the following items (please create a folder for each one):

- Report
- Code + README
- Results + Demonstrative video

Submission will be done via the Assignments of Microsoft TEAMS[1]. **IMPORTANT**: Please make sure that you actually turn in your assignment[2]. If you only upload your files, but do not turn in the assignment, your project might not be considered for evaluation. If you are unsure if your assignment has been turned in correctly, please contact the assistants.

[1]Please, contact the assistants if you encounter problems with TEAMS.

[2]`https://support.microsoft.com/en-us/topic/e25f383a-b747-4a0b-b6d5-a2845a52092b`

Report: The final part of this project consist in writing a report of **max 7 pages**—not counting the references[3]. The report must be written in English and it must describe your work. You have to explain both the idea and the implementation of your solution. However, you should not get into the technical details of the implementation (that is what the well-documented code is for), but rather provide a high-level overview of your implementation. A graphical representation of the control software (for example, in the form of a finite-state machine) is mandatory. Additionally, you have to analyse your results, that is, discuss the limitations and strengths of your approach. To better manage your space, you should keep the description of the problem as short as possible.

The report must be typeset using the template of Lecture Notes in Computer Sciences[4] (LNCS – Springer), available at the TEAMS assignment in the file: Templates_report.zip The template offers a specific page layout, **do not** use any software that changes the page layout (for example the LaTeX package `geometry`).

The format of your report will be that of a scientific article, including abstract, introduction, short description of the problem, description of the solution, results, conclusions and references (a few examples on how to structure your project document are included in the file: Example_scientific_articles.zip, also attached to the assignment).

Code: You have to submit your code following the following guidelines:

- The Lua script that you developed, **commented** and well-structured.
- A README file explaining how to use your code.
- The code should be ready to be tested by us. This means the code should not generate any errors when executed. A common problem is hard-coding file paths that depend on your system. If you need to include these paths, expose them as a global variable and explain in the README what needs to be changed before execution.
- You must implement your own code. **Plagiarism will be strongly penalised**.

Results:

- Create a *csv* file with the results generated by ARGoS for each experiment. Also, include a copy of all your box plots and statistical tests carried out to analyse the performance of your implementations.

[3]You may add as many pages of bibliographical references as you need to support your work.

[4]`https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines`

- Include a video of your control software with the swarm composition that you chose to design your software for at first. You can create a series of images in ARGoS by pressing the round record button before starting your experiment. The generated images can be transformed into a video by the following command:
  `ffmpeg -r 10 -i frame_%010d.png video.mp4`.
  You can also decide to record your screen using a screen capture program. **Do not** record your screen using an external camera (webcam, phone). The resulting video will be several minutes long. Please speed up the video by a proper factor so that the resulting video is at most one minute long. Using the following command, you will speed up the video by 10x:
  `ffmpeg -i video.mp4 -r 60 -filter:v "setpts=0.1*PTS" video_10x.mp4`

3 Final remarks

Apply what you learned — It is very important that you stick to the principles of swarm intelligence: simple, local interactions, no global communication, no global information.

Avoid complexity — Good solutions are elegantly simple in most cases. If your solution is becoming too complex, you are most likely in the wrong direction.

Honesty pays off — If you took inspiration from scientific papers or websites, cite the source and say why and how you used the idea.

Manage your time — This project requires that you run multiple experiments on your own computer, which may be time-consuming depending on your coding skills and the computing power that you have available. Take this into account so that you can deliver a complete project on time.

Cooperation is forbidden — Always remember that this is an individual work.

4 Contacts

Guillermo Legarda Herranz	guillermo.legarda.herranz@ulb.be
Ilyes Gharbi	ilyes.gharbi@ulb.be
Jeanne Szpirer	jeanne.szpirer@ulb.be