Evaluation and Final Project of INFO-H-414
Swarm Intelligence

Second Session

1 July, 2025

1 Modality

The evaluation of the course consists of three parts. To pass the course, you must pass each of the three parts separately with a minimum mark of 10/20. If you pass all three parts, your final mark will be computed as the average of the marks you obtained in each of them.

- **Project.** You will need to solve a swarm robotics challenge and submit a report with your results. The details of the project are provided in Section 2 of this document.
- **Swarm robotics: oral exam.** You will answer questions from Prof. Birattari on the contents of the course concerning swarm robotics. During this part, you might also be asked to answer some questions about your project, should there be any doubts about your methodology or the results you obtained.
- **Optimisation: oral exam.** You will answer questions from Prof. Dorigo on the contents of the course concerning optimisation.

IMPORTANT: if you already passed any of the three parts during the first session, you will not need to pass them again.

1.1 Timeline

- The project submission deadline is **August 15** at 23:59.
- Delay on the submission will entail a penalty of 1 point every 12 hours of delay on the final evaluation of the project. Maximum delay is **August 17**, at 23:59. After this deadline you fail the exam.

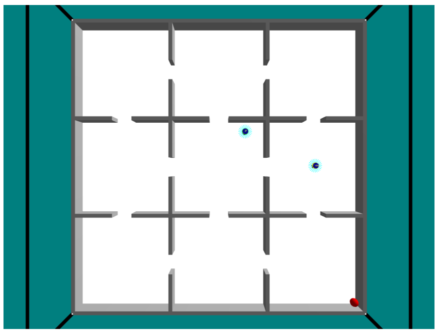
Figure 1: A top view of the arena.

2 Project: *shepherding*

Sheep are flock animals, e.g., they tend to form herds. When shepherds move sheep from one location to another, they do so by moving the whole herd rather than individual sheep. Herders do not always interact with the herd directly but often through trained animals such as shepherding dogs.

In this project, you will need to write control software for a group of shepherding robots. Their task is to move a group of sheep through a series of gates into a target area.

2.1 Problem definition

In this mission, the arena is composed of multiple rooms, see Figure 1 for a top view of arena. The rooms are connected through gates, through which the robots can move from one room to another. Initially, the shepherding robots start in the central room, while the sheep robots are distributed randomly throughout the whole arena.

The mission of the shepherding robots is to move as many sheep into the room with the red LED as possible. The sheep robots react to LEDs in their environment. They will be attracted towards green LEDs, repelled from blue LEDs and they will remain unaffected by any other LED colour.

Each experimental run lasts 6,000 time steps. You are free to reduce this time if the solution you have developed effectively addresses the problem in a shorter time. If you reduce the simulation time, please clarify and justify it in the report.

Goal Your goal is to design a collective behaviour (i.e., a control software) so that the shepherd robots can guide the sheep robots into the target room. The performance is measured by the number N of sheep robots that are within the nest at the end of the scenario (6,000 time steps in ARGoS). While the swarm performs the mission, ARGoS outputs the performance at every time step during the experimental run (in the simulator or in the console, depending on whether the visualization is activated or not—see Section 2.3).

Swarm composition The swarm consists of two different types of robots: sheep and shepherds. The sheep only react to external stimuli, namely green or blue LEDs and obstacles in their proximity. The shepherds will be programmed by you and they will have to guide the sheep into the target area. See Table 1 for a summary of the sensors and actuators available to the shepherds:

Table 1: Sensors and actuators available to the shepherd robots

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

Additional remarks You are given an initial template of control software in the file `shepherding.lua`. This template is the starting point to develop your solution. In the file, we have indicated where you can place your code and also which parts should remain unchanged. Since the same lua script will control the sheep robots and the shepherd robots, you cannot program all robots using the same `step()` function. We have prototyped a functions `shepherd_control()` where you should implement your control software for the shepherd robots.

In the file `shepherding.argos`, we provide you with a swarm of one shepherding robot and twenty sheep robots. You will need to choose the number of shepherding robots that you want to design your control software for. In order to set the right number of shepherding robots, find the lines with the following format (usually lines 66 and 77):

```
<!-- You can play with the number of shepherding robots 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.

2.2 Objective

The objective of this project is to design, implement and test control software for the shepherding robots that allows them to collectively move a number of sheep robots from different places in the arena into the target room. However, this project is not only about optimizing the performance of the swarm. Instead, we are also interested in seeing solutions that make use of the swarm robotics principles that you learned during the course. You are also asked to investigate two properties of the robot swarm: flexibility and scalability. 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 in the form of boxplots, where each box corresponds to the ten repetitions. Please include both the absolute performance (number of sheep in the target room) and the performance per shepherding robot (absolute performance divided by the number of shepherding robots in the mission).

In order to better understand the results that you will receive from your experiments, you should also compare your results against a baseline solution. We provide you with the file `shepherding_with_random_walk.lua` which implements a shepherding behaviour through the use of a random walk. Bonus points can be earned for implementing your own baseline behaviour.

First, in order to test the flexibility of the swarm, you are asked to run your control software (and the baseline) in different arenas. We provide you with four variations of the initial arena: `shepherding_wide_gates.argos`, `shepherding_unbounded.argos`, `shepherding_no_obstacles.argos` and `shepherding_4x4.argos`. To conduct the experiments, you are asked to test the baseline and the control software you have produced without further modification in all five arenas. Try to answer the following questions when analysing the results: Is the performance higher or lower than in the original arena? What could have caused this difference in performance? Does your software perform better or worse than the baseline? Why, or why not?

Second, in order to investigate the scalability of the swarm, you will again test the control software that you developed without further modifications. In the original arena, test the baseline and your control software for twice as many and half as many shepherding robots as you selected to design your control software for. In case that you designed your control software for an odd number of shepherding robots, you can choose whether to round up or down when assessing half the number of robots. For each number of shepherds, please investigate the performance for 20, 40 and 60 sheep robots in the environment. 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? What are the possible reasons that would explain the observed development of the performance?

Bonus points are awarded for simple and reactive controllers accompanied by an in-depth analysis of the results. Remember to follow the principles of swarm robotics and apply what you learned during the course. 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.

2.3 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 shepherding.argos     # Run the experiment
  ```

If the usual ARGoS GUI appears, you are 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. On line 52 of `shepherding.argos` you will see that the Lua controller has a section <params />. If you want to experiment with different controllers, just change the script attribute in line 52 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 12 of `shepherding.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 on line 9 of `shepherding.argos`. Experiment with the values, because the best setting depends on your computer.

2.4 Deliverables and evaluation

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.

Report: The final part of this project consists 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

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

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

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

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 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.
- Include a video of your control software with the original swarm composition and the environment that you chose to design your software 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`.

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

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 Contacts

Marco Dorigo	for general questions
Mauro Birattari	for general questions
Guillermo Legarda Herranz	for swarm robotics
Jeanne Szpirer	for swarm robotics