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

First Session

3 April, 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.

If you fail either of the oral parts, you will be asked to retake that part in the second session. Should you fail the project, you will need to submit a new project for the second session.

1.1 Timeline

- The project submission deadline is **May 18** 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 **May 20**, at 23:59. After this deadline you fail the exam.

2 Project: *constrained foraging*

Foraging is a collective behaviour where a set of items are retrieved from a source location and brought into a target location (often called the nest). In constrained settings, a group of agents need to coordinate themselves in order to forage effectively. Foraging is a fundamental collective behaviour that can be found in many natural systems. For instance, ants are performing foraging by transporting leaves and other resources back to their nest.

In this project, you are asked to provide a robot swarm with the capability of transporting cylindrical objects (from here on called items) into a nest area. The items are initially randomly distributed on one side of a rectangular arena. The nest area is located on the other side of the arena. Because a robot can only transport one item at a time, a single robot cannot solve this task effectively. However, through cooperation of the individual robots, the swarm can transport a larger number of items.

2.1 Problem definition

The robot swarm operates in a scenario composed of a rectangular arena. We provide four different layouts of the arena (see Figure 1), but many characteristics are shared between the four layouts.

In the bottom part of the arena, several items are randomly distributed. The items can be seen by the robots using their omnidirectional camera. A robot can physically pick up an item using its gripper. The position of the robots and items are chosen at random each time you start an experiment.

The arena also contains two grey areas (with different shades of grey). The dark grey area, which is located in the top part of the arena, is the nest area. The robots should transport the items from the bottom part into the dark grey area. The light grey area is located between the items and the nest area. It can be used as a cache or as a cue for navigation. The shades of both grey areas change randomly every time you start an experiment, but the nest area will always be darker than the cache area. Additionally, the arena will also contain a light source that is centred in the middle of the arena.

Three of the four possible layouts also contain walls, that restrict the way that the robots can move from one side of the arena to the other. If walls are present, then the cache area will be located between the walls.

Goal Your goal is to develop a behaviour that lets the robots transport as many items as possible into the foraging area. The performance is measured by the number N of items that are within the nest area at the end of the scenario (6,000 time steps in ARGoS). After the scenario finishes, ARGoS outputs the performance value achieved by this run (in the simulator or in

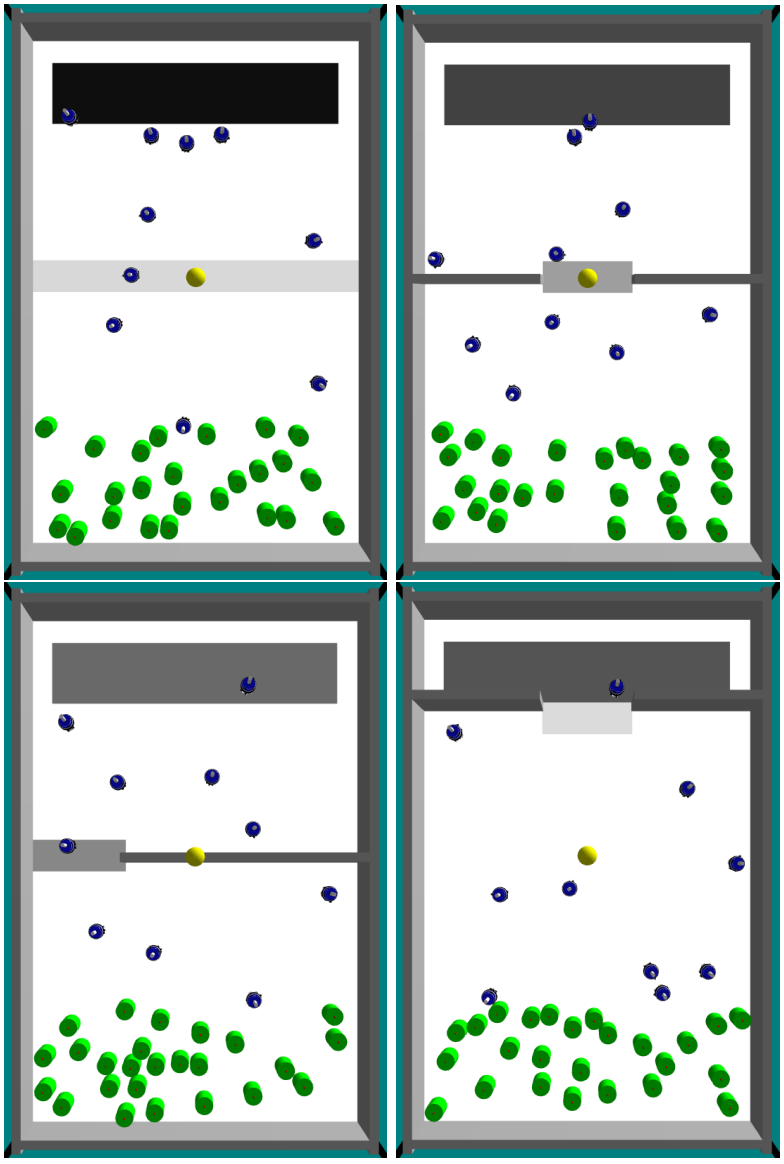

Figure 1: A top view of the four different layouts of the environment.

the console, depending on whether the visualization is activated or not—see Section 2.3).

Swarm composition Originally, the swarm consists of 10 homogeneous robots. See Table 1 for a summary of the sensors and actuators available.

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.light`	`robot.gripper`
`robot.motor_ground`	`robot.turret`
`robot.id`	

Additional Remarks

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

In the files `foraging_s*.argos`, we provide you with a swarm of 10 robots. You will need to choose the number of robots that you want to design your control software for. In order to set your desired number of robots, find the lines with the following format (usually line 94):

```
<!-- You can play with the number of foot-bots 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 into the arena.

Figure 1 shows a top view of the four different possible layouts of the environment. In Figure 2, two different floor colours are shown. If you want to keep the same shades (only for testing, the solution must operate with changing floor colours) set the reset_all property to "false" (line 23 of the `foraging_s*.argos` files).

2.2 Objective

The goal of this project is to design, implement and test a robot controller that allows the swarm to collectively forage the items distributed in the arena. 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

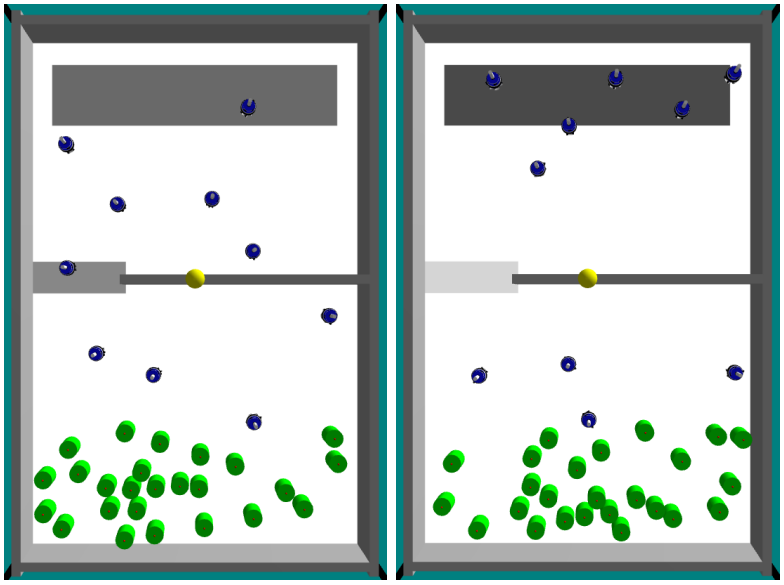

Figure 2: Two different resets of the same layout

principles that you learn during the course. You are also asked to investigate two important properties of robot swarms: 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 each of 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 test the **flexibility** of the swarm, we provide you with four different scenarios, differing in the layout of the arena. You have to select one (e.g., `foraging_s2.argos`) and develop the software for this scenario. Afterwards, you will test the same control software without any further modifications in the other three scenarios. When analysing the results, try to answer the following questions: Is the performance higher or lower than in the original scenario? What could have caused this difference in performance? What could you have done differently to achieve similar performances in all scenarios?

In order to investigate the **scalability** of the swarm, you will select the same scenario that you used to design the control software and test it 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 robots by 2-5 is a good rule of thumb. However, remember to switch off the visualisation of ARGoS (see Section 2.3) when you experiment with large swarm sizes. Otherwise, your experiments will take very long.

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.

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 foraging_s1.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. On line 57 of `foraging_s*.argos` you will see that the Lua controller has an empty section <params />. An example of how to set the script is on line 60 of the same file. Just change line 57 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 on line 12 of `foraging_s*.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 `foraging_s*.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].

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

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

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 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 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:

[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`

- 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`.
  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