LINFO1361: Artificial Intelligence
Assignment 3: Multi-Agent Systems

Amaury Fierens, Harold Kiossou, Achille Morenville, Alice Burlats, Eric Piette
April 2025

Guidelines

- This assignment is due on **Friday 16th May 2025, 18h00**.
- *No delay* will be tolerated.
- *Document* your source code (at least the difficult or more technical parts of your programs). Python docstrings for important classes, methods and functions are also welcome.
- Copying code or answers from other groups (or from the internet) is strictly forbidden. Each source of inspiration must be clearly indicated.
- Source code shall be submitted on the online *INGInious* system. Only programs submitted via this procedure will be graded. No program sent by email will be accepted.
- Respect carefully the *specifications* given for your program (arguments, input/output format, etc.) as the program testing system is *fully automated*.
- The answer to questions must be given by filling in the latex template provided. The final file must be submitted on *gradescope*. No report sent by email will be accepted.
- Nothing must be modified in the template except your answer that you insert in the *answer* environments as well as your names and your group number. The names are provided through the command *students* while the command *group* is used for the group number. The dimensions of *answer* fields *must not* be modified either. For the tables, put your answer between the "&" symbols. Any other changes to the file will *invalidate* your submission.
- To submit on gradescope, go to `https://gradescope.com` and click on the "log in" button. Then choose the "school credentials" option and search for *UCLouvain Username*. Log in with your global username and password. Find the course LINFO1361 and the Assignment 3, then submit your report. Only one member should submit the report and add the second as group member.
- Check this link if you have any trouble with group submission `https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members`

Deliverables

- The following files are to be submitted:
  - `report_A3_group_XX.pdf`: Answers to all the questions in a single report following the template given on moodle. Remember, the more concise the answers, the better. This deliverable should be submitted on gradescope as mentioned before.
  - The file `AntStrategy_concurrent.py` containing your Python 3 implementation of the strategy where ants does not collaborate.

- The file `AntStrategy_collaborative.py` containing your Python 3 implementation of the strategy where ants does collaborate with each other using pheromones.
- The file `AntStrategy_smart.py` containing your Python 3 implementation of the strategy where ants are the smartest possible to achieve their goal in a minimum of time.

Anti plagiat charter

As announced in the class, you'll have to electronically sign an anti plagiat charter. This should be done ***individually*** in the **INGInious** task entitled *Assignment 3: Anti plagiat charter.* Both students of a team must sign the charter.

Ant colony: Introduction

During lectures, you have explored the domain of Multi-Agent Systems (MAS) encompassing both Non-Cooperative and Cooperative game theory. In this assignment, you will have the opportunity to implement a multi-agent system in the context of a colony of ants that seek to gather all food from a predefine maps as quickly as possible. To be able to compare the efficiency of cooperative and non-cooperative strategies, you will have to implement three different strategies for the ants to collect food. These strategies will be discussed further in the Assignment.

To implement all your strategies you will have access to limited information. Indeed, instead of each ant having access to the whole map, each ant will only have access to the perception it has of its immediate surroundings. In the lectures, this corresponds to a environment with hidden information.

1 Framework Description

To implement your strategies, you will have to deal with the framework we have designed for you. A little diagram of it is available at Figure 1.

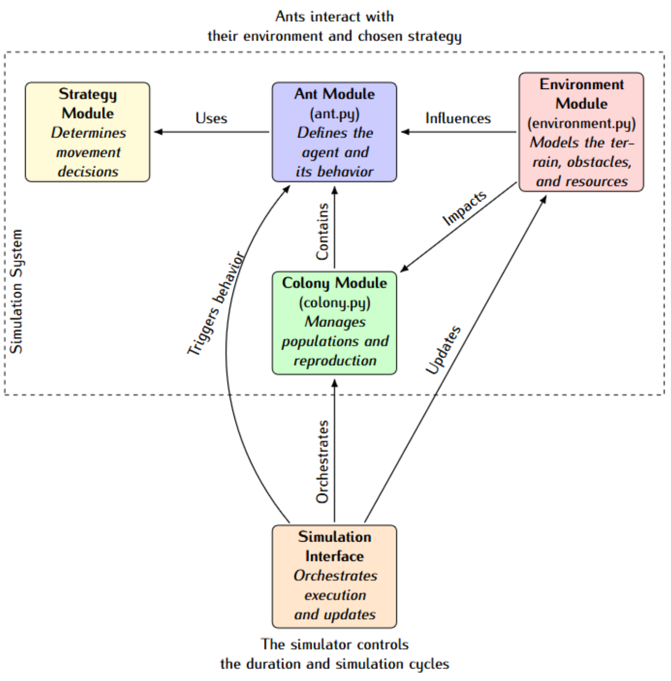

Ants interact with their environment and chosen strategy
Strategy Module Determines movement decisions
Uses
Ant Module (ant.py) Defines the agent and its behavior
Influences
Environment Module (environment.py) Models the terrain, obstacles, and resources
Simulation System
Contains
Impacts
Triggers behavior
Colony Module (colony.py) Manages populations and reproduction
Updates
Orchestrates
Simulation Interface Orchestrates execution and updates
The simulator controls the duration and simulation cycles

Figure 1: Framework interactions

We do not provide details about the Colony Module, as it is not very useful for you. However, for the Ant Module, the Strategy Module and the Environment Module, along with the Simulation Interface, more explanations are given in Section 2.

Figure 2 shows how our Graphical User Interface (GUI) represent ants and pheromones for a 100x100 environment where 4 food cluster are located are each corner of the map. By default, ants are in brown, and in orange when they carry food. The food squares are also in orange, while the pheromones that lead to the colony are in green and the pheromones that lead to the food are in white.

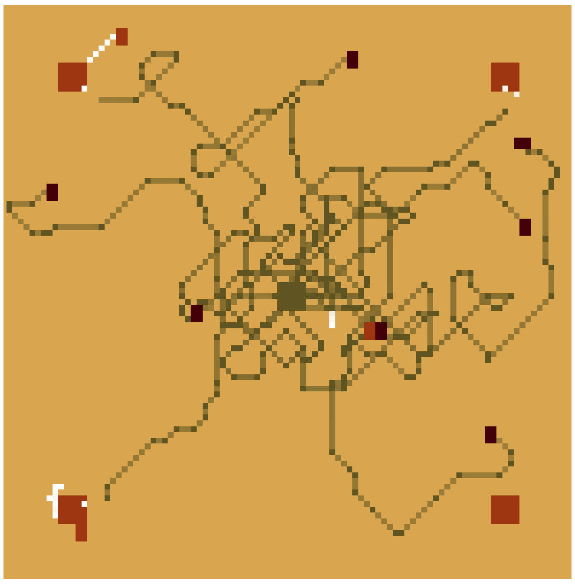


Figure 2: GUI example

2 Your work

In this framework, as described before, you will have to implement three different strategy modules. These modules will be used by the ant modules to simulate their behavior.

- **The first strategy module** is a non-cooperative one, in which each ant acts independently. Their objectives is to collect food located outside of the colony and bring it back. Rather than sharing information, rely solely on its own memory, which you are free to implement. ***Note:*** *For this agent, pheromone will not be available, as all related methods will be deactivated.*
- **The second strategy module** is a cooperative one, where ants collaborate to collect food. They use pheromones to communicate information about the location of food and the colony. However, these agents cannot use any form of memory to store location information (e.g., the position of food or the colony).
- **The third strategy module** is a smart agent, who takes the best of both previous strategies, along with possible improvement that you may add. These improvements can, for example, involve implementing concepts covered during the lectures.

Hint: *You could for instance consider: (1) Methods for searching the shortest path (in the case of ants with memory), (2) Methods for balancing between Intensification and Diversification, (3) Smart ways of using pheromone trails information, (4) Using information about concentration of ants at a given spot, ...*

2.1 Implementation

Your job is to create 3 files, one per strategy. Each one of them should extend the `AntStrategy` class from `ant.py`. In each of those files, the function that you should implement to be able to compile is `decide_action(self, perception: AntPerception) -> AntAction`. It takes as input the `Ant` object itself, and the `AntPerception` object, and it outputs an `AntAction` object. Here is a description of each of these classes to allow a better understanding of the interactions between them:

`Ant`

The `Ant` class represents an autonomous agent in the simulation. Each ant has a position (x, y) (it is (0,0) at the point where it spawned), a direction, and follows a specific strategy to decide its actions. It can move, turn, pick up, and drop food.

In your **first agent**, the ants **don't use the pheromones** to communicate because no communication is allowed. However, ants dispose of a memory and can therefore keep track of the locations of several things (obstacles, colony, food, ...).

In your **second agent**, each ant should interact with its environment by emitting **pheromones**, with separate levels for **home** and **food trails**, which are set to decay .

Your **third agent** can, for instance, be a mix of both previous agents with improvements.

The behavior of the ant is defined by the function `decide_action(self, perception: AntPerception) -> AntAction` of the class `AntStrategy`. The ant tracks its movement, food collection, and adapts based on pheromone signals or other information such as memory depending on the agent.

`AntPerception`

The `AntPerception` class defines what an ant can perceive in its environment. It provides access to sensory information, including visible terrain, pheromone levels, and nearby ants.

- **Visual Perception:** The ant can detect terrain types (`COLONY`, `FOOD`, etc.) in its surroundings, stored in `visible_cells()`.
- **Pheromone Detection:** The ant can detect levels of pheromones in nearby cells using `food_pheromone()` and `home_pheromone()` attributes.
- **Nearby Ants:** The `nearby_ants()` attribute keeps track of other ants within perception range.

Please note that for all these attributes, the coordinates that you will get back are **related to the ant**, not the environment, as shown in Figure 3.

Please note also that the ant perceptions are defined by their field of view (FOV). It is a cone with the edge at the extremity of the ant square, opposite to the gaze direction. It has an angle of 120°and a radius of 4 squares. In Figure 4, the blue squares are the squares that the ant can see.

The class also includes helper methods:

- `can_see_food()`: Checks if food is visible in the ant's surroundings.
- `can_see_colony()`: Checks if the colony is visible in the ant's surroundings.
- `get_food_direction()`: Determines the best direction towards the nearest food source using the `can_see_food()`.
- `get_colony_direction()`: Determines the best direction towards the colony using the `can_see_colony()`.

These functions allow the ant to make informed decisions based on its perception of the environment.

This class is the most useful one for your `AntStrategy` implementation, as it contains the majority of the methods that will be useful for you to determine your next action.

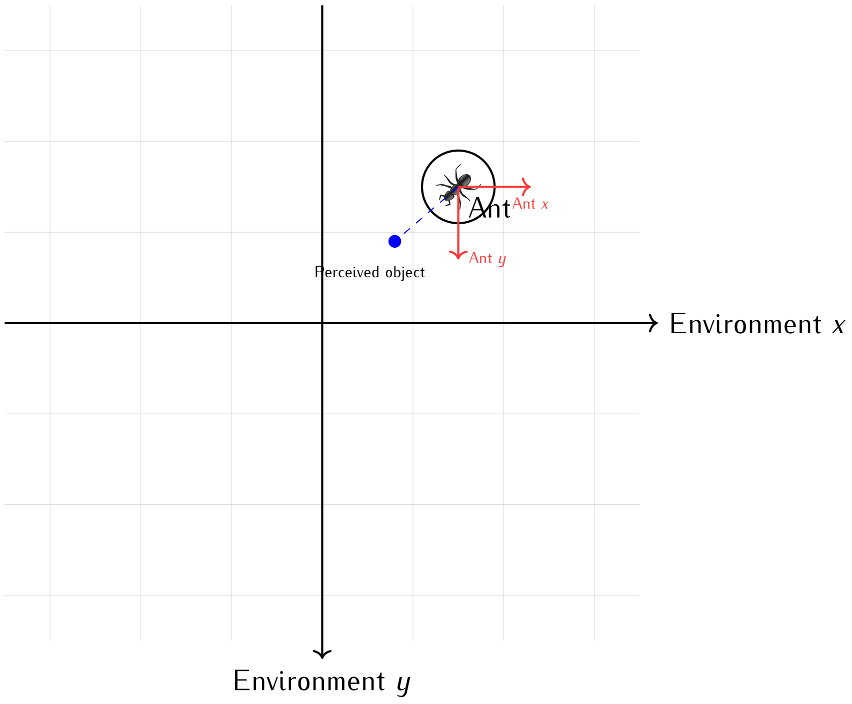


Figure 3: Illustration of ant-relative coordinates in a global environment.

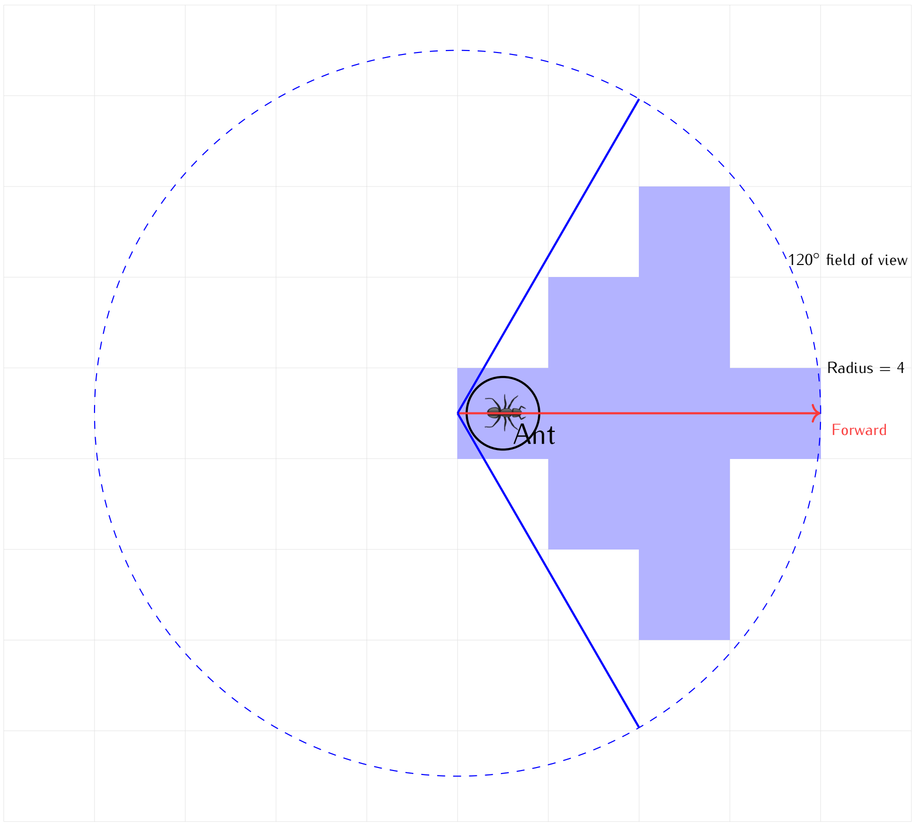


Figure 4: Blue squares are the squares considered by the algorithm to be part of the ant's FOV. The field of view starts from the left edge of the ant's square.

`AntAction`

The `AntAction` class defines the possible actions an ant can take in the simulation. These actions allow the ant to navigate, interact with food, and communicate through pheromones.

- MOVE_FORWARD: Moves the ant one step forward in its current direction.
- TURN_LEFT: Rotates the ant 45°counterclockwise.
- TURN_RIGHT: Rotates the ant 45°clockwise.
- PICK_UP_FOOD: Allows the ant to pick up food if it is standing on a food source.
- DROP_FOOD: Drops the carried food at the current location.
- DEPOSIT_HOME_PHEROMONE: Releases a pheromone signal indicating a path towards the colony.
- DEPOSIT_FOOD_PHEROMONE: Releases a pheromone signal indicating a path towards a food source.
- NO_ACTION: The ant remains idle without performing any action.

These actions allow the ant to navigate its environment, gather food, and coordinate with other ants using pheromones. In particular, as described in TURN_LEFT and TURN_RIGHT, the ant can rotate from 45°as it can also move diagonally.

2.2 How to run

Once you will be done with the implementation part, you will probably want to run the simulation. To do this, below is some information required about the Simulation Interface and the Environment Module.

Note: *If you want a complete description of all the command-line arguments, the GUI mode, the Simulation mode, the creation of your own environment, you have access to the* `command_line_args.md` *file.*

Simulation Interface

To emulate the environment and the ants, two possibilities are given to you:

- **simulation.py**: If you want to emulate the whole execution at full speed, like in the conditions of evaluation, you should use this file. To use it, the command line is:

```
$ python3 simulation.py --env path/to/env --strategy-file name_of_file.py
  --time-limit nb_sec --max-steps nb_steps (--quiet)
```

- **gui.py**: If you want to visually see the execution of each ants, to be able to see what goes well and what goes wrong during emulation, you should use this file.

```
$ python3 gui.py --env path/to/env --strategy-file name_of_file.py
  --time-limit nb_sec --max-steps nb_steps
```

Here, the --quiet argument is optional here. If not given, the program will return information about the current emulation.

The path/to/env must be the path to an environment file as described at section 3.2.

2.2.1 Environment Module

The Environment Module is responsible for managing the simulation world in which the ants operate. It defines the grid-based space, as well as the locations of key elements such as the colony and food sources. While you are not required to modify this module, understanding the environment file format is crucial for testing and designing effective agent behaviors. To do so, we invite you to refer yourself to either section 3.2 or to the document `command_line_args.md`.

2.3 Open questions

In addition to implementing the three strategies, your report should include your thoughts on possible improvements that could be made under three different conditions.

Discuss improvements given what you have seen during this year when:

1. You have access to the AntStrategy module as for the implementation, but you also have all the information about the obstacles and the food locations.
2. You have access to all the ants, and you can also create coalitions of ants (like the ants are drones). You don't have any information about the environment (obstacles, food location). You still have the ants sensors.
3. You have access to all the ants, but you can also create coalitions of ants. You do not have access to the food locations, but you do have access to the ants sensors and to the precise location of the obstacles on the map.

3 Evaluation method

To evaluate the performance of your strategies, we will follow the given method of grading:

First, for your basic agents (the non-cooperative and the cooperative one), an evaluation will be performed on a "baseline" environment, given in file `XX_square_diag.txt` that we will provide. If they are able to solve this environment in a limited amount of time and moves, then you will be granted **2 points** per agent that passes this basic environment.

Reminder: *Please note that for the first agent (the non-cooperative one), you won't have access to any pheromones as all the methods related to them will be deactivated.*

Then, your smart agent will be tested on a total of 10 different environments. We will provide 5 of them to you, but 5 will remain hidden. The 5 environments provided will be located in the `envs` folder. The 5 hidden environment will have similar degree of difficulty to those provided. The difficulty for the 10 instances will come from either the number of ants that are given to solve it, the size of the environment, the obstacles and the size and location of the food spots. For each one that you passes (again in terms of number of moves and time), you will be rewarded **1 point**.

Note: *An instance will be considered solved if 90% of all the food has been collected and stored in the colony in the time limit and max number of steps described in the environment.*

The three open questions will be rewarded **2 points** each. The evaluation will be performed on the feasibility and the efficiency of your methods, along with the usage of concepts seen in the lectures. You will be rewarded more points if you use concepts seen in the lectures.

3.1 Summary of grading

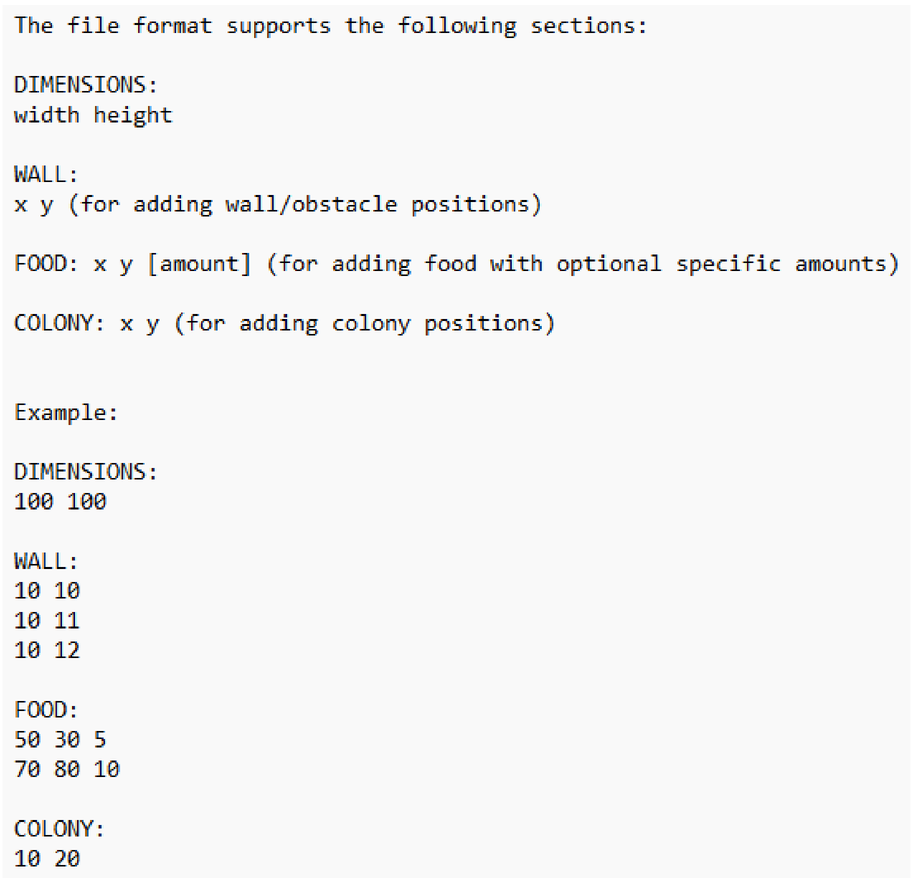
- **Implementation** [14/20]
  - Non-cooperative agent [2/20]
  - Cooperative agent [2/20]
  - Smart agent: 1 point/environment [10/20]
- **Report** [6/20]
  - Open question 1 [2/20]
  - Open question 2 [2/20]

– Open question 3 [2/20]

3.2 Create your own Environment

To be able to better assess the performances of your smart agent, we allow you to create your own environment to test it, with all the constraints you want to add in terms of obstacle and number of food squares.

To do it, you can do several things. First, you can eventually create a new environment file. It should be a .txt file and should be structured as described in Figure 5.



```
The file format supports the following sections:

DIMENSIONS:
width height

WALL:
x y (for adding wall/obstacle positions)

FOOD: x y [amount] (for adding food with optional specific amounts)

COLONY: x y (for adding colony positions)


Example:

DIMENSIONS:
100 100

WALL:
10 10
10 11
10 12

FOOD:
50 30 5
70 80 10

COLONY:
10 20
```

Figure 5: Example of environment

Another way to do it is to use the pre-built environment creation tool we provide you. To use it, here is the kind of command line you should use:

```
$ python3 gui(or simulation).py --env maze --strategy-file name_of_file.py
  --width size --heigth size --time-limit nb_sec --max-steps nb_steps
```

The arguments useful for this creation are `--width size` and `--height size`, to set the height and the width of the environment, and `--env maze` will force the program to generate a random environment with walls and food. The sizes must be integers.

Note: *If you want a complete description of all the command-line arguments, the GUI mode, the Simulation mode, the creation of your own environment, you have access to the* `command_line_args.md` *file.*