MAS: Activity 1 – Predator - Prey with communication

Alexandru Sorici
08.03.2021

The Problem: a classic predator - prey scenario. The environment is a rectangular room with $m \times n$ tiles. Obstacle tiles define the walls of the map.
The map contains *num_pred* predator agents and *num_prey* prey agents. By default there are two predator and one prey agents on the map. All agents are spawned randomly on the map, ensuring that all prey agents are at a Manhattan distance of 3 or greater from any predator agent. Furthermore, each agent of the same type (i.e. predator or prey) is at a Manhattan distance of at least 2 from another.

The predator agents have the ability to send messages to other predators that are known to them. Predators can only send messages to other predators they perceive nearby, or which they *remember* from previous encounters.

The purpose of the game is for the predator agents to eat all prey agents on the map. The predator agents have to show two abilities:

- the ability to *assume roles* and divide the exploration of the map among themselves
- the ability to *use communication* to improve their strategy

Work on the predator agent strategy in an incremental fashion and compute the average number of steps taken to finish the game (over different initializations) for each of your increments to show improvements.

In doing so consider the following specifications.

Specifications

- the agents have four operations: *UP*, *DOWN*, *LEFT*, *RIGHT*.
- prey agents can perceive tiles at a maximum Manhattan distance of 2.
- predator agents can perceive tiles at a maximum Manhattan distance of 3.
- the agent's perception is modelled as a `MyPerceptions` structure, containing information about its current position, obstacles, nearby agents, as well as **the set of messages received from other agents**.
- two agents **can** be located in the same tile
- predator agents **are allowed to use memory** and **communication**.
- prey agents are reactive - they employ a random walk strategy, trying to stay away from predators if they perceive them.
- a prey agent is considered killed in the following conditions:
  - there is at least one predator at a Manhattan distance of 1 to the prey agent
  - there are at least two predators at a Manhattan distance of 2 to the prey agent

To Do 1: Implement in `MyEnvironment` the code for sending perceptions to predator agents, as well as the management of messages. Messages sent at one step will be delivered in the next step.

To Do 2: Design and implement a cognitive behavior, involving communication with other agents, for the predators, showing that this leads to an improved behaviour for both: **map exploration** (don't stay in the same place, explore different regions) and **hunting down prey** (converge quickly on prey).

Note You are allowed to modify the constructor of the Predator Agents to give them the total number of predator agents, as well as the dimensions of the map.

Evaluation Method Create **two types** of predator agents: ones that **do not** use communication and ones **that do**. Use the following procedure:

- Create an environment with a grid size of 10x15 (10 lines, 15 columns), with 4 predator agents and 10 prey agents.
- Run 20 instances of the game in which predator agents **do not** use communication (they **only** use their exploration and prey convergence strategies). Compute **the average number of steps it takes to finish the game**.
- Run 20 instances of the game in which predator agent **use communication** (alter their response strategy accordingly). Compute the average number of steps it takes to finish the game and compare this to the case with no communication.

Helpful pointers:

- An `AgentMessage` contains a sender, a destination and some content. The sender and the destination are `AgentID` instances. The content can be whatever you want.
- Generate necessary `AgentID`s using the static method `AgentID.getAgentID(Agent)`.
- For predators, `response` should return an instance of `SocialAction`. "Social actions" contain the physical action to perform (movement) and also messages to be sent to other agents.
- Messages from the returned `SocialAction` instance must be kept from one step to another in `MyEnvironment.messageBox`. **Careful:** don't store messages from the current step and from the previous step simultaneously in the `messageBox` (i.e. empty the messageBox after you have delivered all the messages from the previous step).