Assignment 1: Implementing the Tileworld Simulation

CSE5022 Advanced Multi-Agent Systems

DDL: 23:59, Wednesday, April 17, 2024

1 Overview

In this assignment, you have to develop some aspects of the **Tileworld simulation** introduced in [1] to evaluate the behaviour of "robot" agents navigating in a simulated dynamic environment, considering the **Belief-Desire-Intention (BDI)** paradigm. For this purpose, you are required to use the open-source Repast Symphony Simulation Multi-agent Toolkit[1], implementing the appropriate classes in *Java* language (considering Object-oriented programming principles).

2 Objectives

- Design and deploy a Multi-Agents System considering a dynamic environment where the agents are situated.
- Implement internal agents' properties, considering notions of Autonomy, Reactivity and Pro-activeness.
- Develop required agents' methods, following the BDI paradigm [2]. For this reason, *Belief* of the world should be enabled by "sensing the environment", while Goals or *Desires* are frequently monitored. Moreover, appropriate planning to achieve these goals or *Intentions* (Figure 1) must be ensured.
- Produce simulation results, using Repast features that enable analysis of agents' performance.

3 Requirements

1. In the design stage, you need to identify the essential components for each agent, following the BDI paradigm.

[1] https://repast.github.io/

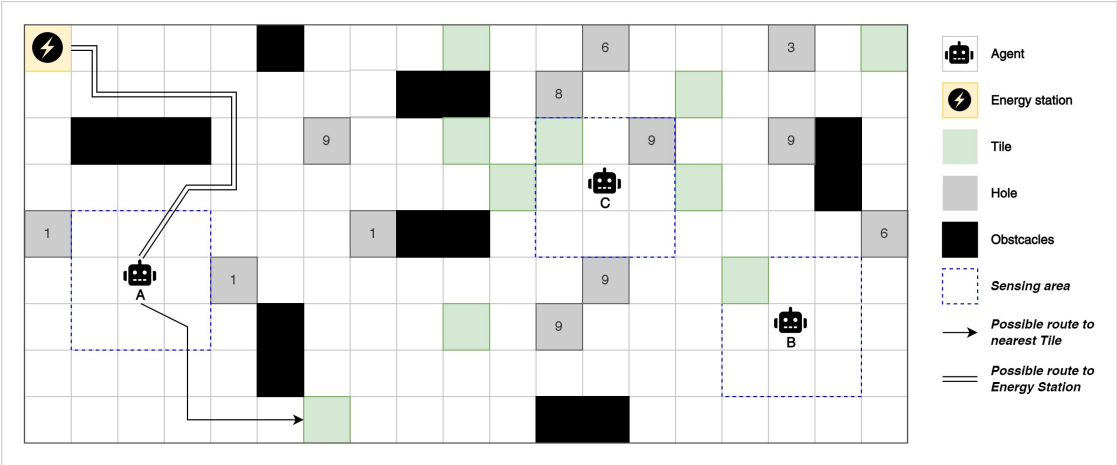

Figure 1: Tileworld simulation

2. In the implementation stage, each **Agent** has the main *Goal* of achieving the maximum score while maintaining its resources. Therefore, they should be capable of:

   - Exploring the environment *moving randomly*, while looking for potential **Tiles** to be collected, *sensing* nearby objects in a given *radio*.
   - At each time step, the agent should monitor the *energy status* to make some decisions. *Initial energy level* should be set to 100 units (%), and *decreases* 1 unit *if the agent moves from 1 cell to another*
   - Once some **Tile(s)** is/are detected, the agent can *Pick up* the nearest **one, by one**, and store them into a *List*. When this action is carried out, the Tile should be removed from the environment (context).
   - Once some **Hole(s)** is/are detected *and* the agent has *available tiles* (in a List), the agent can *Place/Fill* that hole and obtain the reward provided by it. The agent's score *increases* the units provided by the Hole's rewards. Hole's reward is assigned randomly when the simulation starts, with a random value between 0 and 10. Moreover, *if more than one hole is detected*, the agent must fill the *nearest, with the highest score*.
   - The agent should maintain the location of the nearest *Energy Station*. In case the energy level is too low, the agent should recharge it before continuing operations. A threshold triggering this action can be set at initialization (e.g., recharge when energy is below 20).
   - The decisions analyzed and made, at each timestep, involve: Picking up Tiles and Filling Holes as long the agents have enough energy. Furthermore, the movement to execute these actions can be influenced by the presence of **Obstacles**.
   - A key characteristic of the "Tileworld" environment is its dynamism, meaning that it changes certain time steps, moving dynamically the position of Tiles,

Holes and Obstacles *randomly*. For example, considering a simulation of 1000 time steps, every 100 time steps the environment (context) is rebuilt. **Only agents' information should be preserved, including current location, energy level and score**. This approach forces agents to reconsider the planning and decision-making every time the environment changes.

3. Finally, in the simulation stage, some **Parameters** and **Results** should be configured:

   - All parameters should be set using the simulation Runtime GUI, including the environment size (width, height), agents amount, tiles amount, holes amount, obstacles and energy stations amount. Refer to Repast Symphony documentation [2].
   - **Results:** You have to enable Data collection in the simulation interface [3], recording the score achieved by each agent in each tileworld setup using a .csv file.

4 Hints

The **Agent, Tile, Hole and Obstacle** classes are provided, but remember that Agent's class is incomplete, you may use them as a reference to build your program. Consider that the `Context` class provides access to the Objects in the simulation, and `@ScheduledMethod` specify when to trigger any custom method.

5 What to Submit

1. A report in PDF format describing how the agents follow the BDI paradigm in the Tileworld simulated environment. Please write it with as much detail as possible using screenshots of the simulation runtime GUI and code, illustrating the solution step by step.

2. All source code files. It will be evaluated under different parameters set by the user in the simulation Runtime GUI, as requested in Section 3.

Pack all files into `SID_NAME_A1.zip`, where `SID` is your student ID and `NAME` is your name (e.g., 11710106_ 张三 _A1.zip).

6 References

[1] Martha E Pollack and Marc Ringuette. "Introducing the Tileworld: Experimentally evaluating agent architectures". In: AAAI. Vol. 90. 1990, pp. 183–189.

[2] https://repast.github.io/docs/RepastReference/RepastReference.html
[3] https://repast.github.io/docs/RepastJavaGettingStarted.pdf

[2] Michael Wooldridge. Reasoning about rational agents. MIT press, 2003.