Assignment 1: Tileworld

CSE5022 Advanced Multi-Agent Systems

DDL: 23:59, March 30, 2025

1 Overview

In this assignment, you must develop a tile world simulation to evaluate the behavior of "robot" agents navigating a dynamic environment, considering the **Belief-Desire-Intention (BDI)** paradigm. For this purpose, you are required to use the open-source Repast Simphony Simulation Toolkit[1], implement the appropriate classes in *Java* language (considering the principles of object-oriented programming).

2 Objectives

1. Design and deploy a Multi-Agent System where the agents are situated in a dynamic tile world environment.
2. Implement internal agent properties, considering notions of Autonomy, Reactivity and Pro-activeness.
3. Develop methods that allow the agents to follow the BDI paradigm [1]. For this reason, *Belief* of the world should be enabled by "sensing the environment", *Desires* should be frequently monitored, and appropriate planning to achieve *Intentions* (Figure 1) must be implemented.
4. Produce simulation results, using Repast features that enable analysis of agents' performance.

3 Model

The agents are robots in a tile world that has been degraded and is now filled with holes! However, some tiles are left around in random locations. The robots have seen this and aim to fill the holes with tiles to prevent people from falling in them and getting hurt. Some holes are along critical paths and thus filling them first is more important. A reward value dictates the importance of each hole. Each time a robot puts a tile in a hole, it receives

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

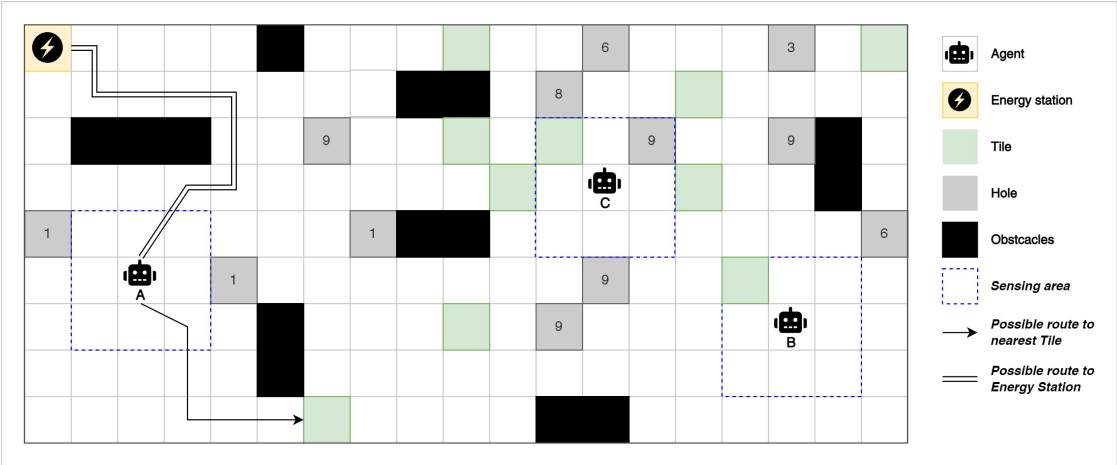

Figure 1: Tileworld simulation

points equal to the reward value of that hole. Thus, to maximize their usefulness, the robots should try to fill holes that give a high reward when given the option to choose. Moving around and filling holes is a tough job, which makes the robots' battery decrease quickly! Specifically, each time they move from cell to cell, the robots lose 1% of battery. Conveniently, the tile world has energy stations that robots can use to refill their energy levels. Still, the robots need to be careful not to put themselves in a situation where their current battery level is insufficient to reach the closest station. Besides holes, tiles and energy stations, the tile world contains some barriers that prevent the robots from going over them, forcing them to go around. Finally, the tile world is experiencing an intense hurricane season, resulting in strong winds that frequently cause the location of the tiles and barriers to change! Our poor robots need to try extra hard to fill the holes.

4 Requirements

Your implementation must produce agents capable of:

1. Exploring the environment by *moving randomly*, while looking for potential **Tiles** to be collected through *sensing* nearby objects in a specified *radius*.

2. Monitor their *energy status* and make pro-active decisions about moving to an energy station. The initial *Initial energy level* of all agents should be set to 100 units (%), and the energy level should *decrease* 1 unit *when the agent moves to a neighbor cell.* To achieve the above, the agent should maintain the nearest *Energy Station* location. If the energy of a robot falls below a configured value (e.g., 20%), the robot should immediately head to the nearest station to charge.

3. Once a **Tile** is detected, the agent can move towards it, *pick it up* and store it locally. The Tile should be removed from the environment (context) upon this action. Each agent can only hold at most 1 Tile at the same time.

4. Once a **Hole** is detected *and* the agent has an *available tile*, the agent can move toward it, *place* the tile in that hole and obtain the provided reward. The agent's score *increases* by the reward provided by the Hole. The Hole's reward is assigned when the simulation starts, with a random value between 0 and 10. As soon as the Hole is filled with a Tile, it should be removed from the environment, and a new Hole as well as a new Tile should be randomly created in the environment.

5. The decisions analyzed and made, at each timestep, involve: Picking up Tiles and Filling Holes as long as the robot has enough energy, otherwise heading to the energy station. If no objectives are found in the sensing area (tiles or holes when the agent has tiles in its list) and the robot has enough energy, the robot should move randomly until it finds something useful to do.

6. Finally, the movement of the robots can be influenced by the presence of **Obstacles**. These obstacles are placed in random cells of the tile world, preventing the robots from moving to these cells and forcing them to go around.

7. To model the strong winds in the tile world, the location of the tiles and obstacles must randomly change. Precisely, these winds are controlled by a timer (e.g, 1000 ticks) after which the logic that changes the location of the above objects in the world must be executed.

In the simulation stage, some **Parameters** and **Results** should be configured:

1. All parameters should be set using the simulation Runtime GUI, including the environment size (width, height), number of agents (robots), available tiles, number of holes, obstacles and energy stations. The Repast Simphony documentation [2] can be very usefull when working on the above tasks.

2. **Results**: You have to enable Data collection in the simulation interface[3], recording the score achieved by each agent in each tile world setup using a .csv file.

5 Clarification

1. A robot (agent) must be in the same cell as the object it's trying to interact with. For example, if a tile is at (4,5) the robot's location must also be (4,5) to pick it up. Same applies to filling Holes and Charging at the Stations.

2. When an agent senses a tile, it must mark it with its ID before moving over and picking it up to prevent other agents from going to the same tile.

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

6 What to Submit

1. A report in PDF format describing how the agents follow the BDI paradigm in the Tileworld simulated environment. Describe the strategy you used for your agents to maximize their score. 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).

7 References

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