CS551K – Software Agents and Multi-Agent Systems 2022-2023

Assessment 3 – Agent Programming Contest

Team Sab-BOT-age Assessment 3 Report

Team members:

ABHIJITH UDAYAKUMAR ADITYA BAWANKULE JOHN MCKENZIE

NIKHILA RAMISETTI SUMAN DEB DUANYUAN XU

Introduction

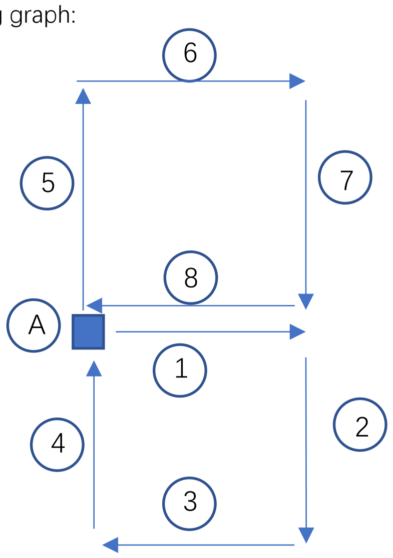
The basic idea of this project is to let the agent discover the maps, reach the position of the dispenser, request the block from the dispenser and successfully deliver it to the goal, then repeat this process to see how many rewards the agent can get from during the limited step.

Main Strategy

Navigation

Process 1: Spider search

The first thing that we do is to adopt a technique which we call spider search. This moving technique can be explained by the following graph:



Spider-search

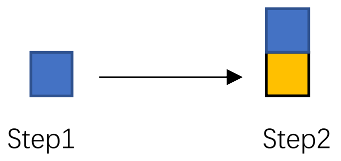
The blue box in the graph indicates the agent. Starting from an initial point A, the agent goes East 6 steps, then goes South 6 steps, then goes West 6 steps, then goes North 6 steps. After the agent discovers one cycle (square), the agent starts to discover another, which shown on the graph is procedure 5,6,7,8. If there is no dispenser, the agent will discover another cycle (square). In addition, the agent will always discover the cycle (square) in the direction of clockwise. We think there is a high possibility that in the cycle (square), the agent will detect some entities along its way.

Process2: Stop the spider-search and get the dispenser

Whenever the agent finds a dispenser, it will immediately stop the spider-search that it previously does and get to the dispenser, the agent will go to the top of the dispenser (which means the go to the exact position the dispenser occupies)

Process3: Move north one step and request the block

This process can be described as the following graph:



The blue box in the graph indicates the agent, and the yellow box indicates the dispenser. The agent moves from step 1 (When the agent is above the dispenser) to step 2 (one step to the north) to request the block from the dispenser.

Process4: Attach the block (b0 or b1)

After the dispenser releases the block (b0 or b1), the agent will immediately attach the block.

Delivery

Process5: carry the block while doing the spider-search for the goal state

After the agent carries the block, the agent immediately switches to the spider-search mode and looks for the goal state.

Process6: Reach the goal state grid

Once finding the goal state, the agent will reach the grid of the place where the goal state occupies.

Process7: Wait and submit the block once the task has been distributed

The agent will check whether the task matches the block it carries, once matches, the agent will submit the task with the task ID.

Process8: Back to spider-search and repeat the loop

After submitting the task, the agent will back to the spider-search state. If it finds the dispenser, the agent will back to process 2 and repeat the loop of process 2 to process 8.

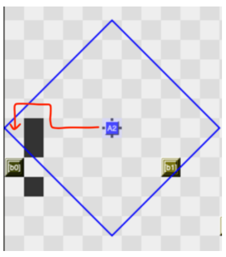
Obstacle/Edge detection:

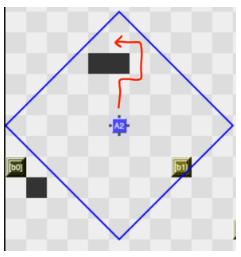
The obstacle and edge detection will be covered every time of the movement of the agent.

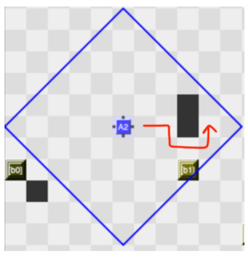
When an agent is moving, it detects an obstacle on its path based on the direction of the agent. The agent will move away from the obstacle before it collides.

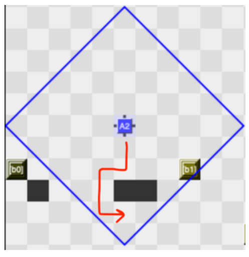
The logic for moving away would be like if the agent detects the obstacles on its way before hitting the obstacles, the agent will always move right. For example, if the agent moves north, and it detects the object to its north, before colliding with each other, then it moves on its east.

Here are the four scenarios when the agent detects the obstacle.

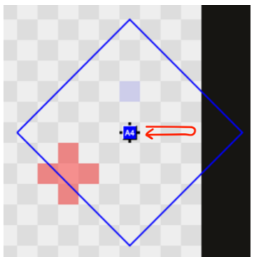

Scenario 1


Scenario 2


Scenario 3


Scenario 4

As for edge detection, the agent will always go to the opposite direction of the edge whenever the edge is detected.



Communication between the agents

- When two agents are within each other' s perception, they add them to their belief base with their local coordinates.

The agents capture their individual local perceptions and broadcast them to each other.
The global coordinates with respect to every agent can be added as an additional percept to their belief base.

Failed Strategies

- **Ladder Search** – Movement of agents in a specific pattern (fixed number of steps in the east, north till the top of the map followed by west and south traversing to the bottom) to explore the maximum possible area of the map for efficient movement.
- **Multiple block connections and agent coordination –** Strategy to attach blocks in all four directions of the agent and efficiently deliver them to the goal state via rotation.
- **Java to Jason action creation –** We could extract and arrange the percepts via the environment (EISAdapter) but could not communicate actions or literals to JASON files.

Additional Strategies (What would we have done if we had more time to develop).

In the following strategies, we wanted to create a global map for every agent with respect to its spawning position and update each agent's belief base with respect to the entities within their perception. The agents once close would broadcast their local perceptions to find an efficient path to complete the tasks (We were successfully able to create the belief bases and communicate it amongst the agents, however, due to time constraints could not implement the idea).
- **A* Heuristic Search**
- **PDDL Implementation** – Post the creation of an entire global map, we wanted to create dynamic instances of problem files with respect to every task generated in the server and a constant domain file.

With a fast-downward.py dependency execute a plan to find an efficient path.