CS551K – Software Agents and Multi-Agent Systems 2023-2024 Assessment 3 – Agent Programming Contest Report

Navigation

Process 1: Random Movement Until Dispenser is Seen.

Initially, the agent moves randomly on the map without storing the location of dispensers. This action continues until the agent encounters a dispenser. Once a dispenser is seen, the agent adjusts its behavior to interact with the dispenser and collect blocks.

Process 2: Reaching dispenser.

The agent detects a nearby dispenser and decides its next action based on the dispenser's position relative to its own. It selects a directional movement, left (west), right (east), up (north), or down (south) to approach the dispenser and collect available blocks.

Process 3: Request and Attach block.

The agent requests blocks from nearby dispensers if it's not already attached to a block. It sends requests in the directions (north, south, east, west) where dispensers are detected. If no dispensers are found nearby, the agent resorts to random movements to continue exploration. When the agent detects a dispenser with an adjacent block, The agent checks if the block at the dispenser is unattached by another agent before attempting to attach to it. If the block is available and not already attached, the agent proceeds to attach to it.

Delivery

Process 4: Searching for the GOAL

When the agent is attached to a block, it attempts to move towards the goal if one exists. It evaluates the goal's position relative to its current location and selects the appropriate movement direction, west (w), south (s), east (e), or north (n), to progress towards the goal. If no goal is specified, it resorts to random movements to continue exploration for the nearest goal.

Process 5: Goal Reached

When a goal is specified, the agent moves towards it by adjusting its direction based on the goal's coordinates. If the goal coordinates are negative, the agent moves west (left) or north (up), and if they are positive, the agent moves east (right) or south (down). The agent updates its internal state to indicate that the goal has been found.

Process 6: Adjusting Agent position.

When the agent reaches the goal, it takes a step back to adjust its position by rotating clockwise if the block is to the right of the goal, or counterclockwise if the block is to the left or below the goal. This ensures that the position of the block relative to the agent is southward upon submission.

Processes 7: Task Submission

The agent verifies any associated tasks. If a task exists, the agent submits it to the environment and then detaches from the block for the next task.

Obstacle detection

The implementation of a strategy to avoid blockages based on obstacle location and agent behavior is crucial as the agents automatically avoid obstacles in their path as they only attach to blocks. However, the primary challenge is when agents are attached to blocks and obstacles obstruct the block's path. In those cases, we rotate the block clockwise if the obstacle is above it and the agent is moving north, or counterclockwise if the block is to the left of the agent. This strategy ensures obstacle-free movement for blocks. For other obstacle scenarios, we rely on our failed_path error handling strategy.

Error handling

1) failed_path:

A failed path error is usually when an agent cannot move to the target location because the agent or one of its attached things is blocked. Our strategy here is to get the parameters, as in the direction in which the error occurs, and to ask the agent to move in a different direction. This strategy should ideally work in resolving the failed path as it changes the course of the agents.

2) failed_forbidden:

Failed forbidden error occurs at the edges of the map when the agent tries to move to a position that is out of the grid/map. Like our previous strategy with handling failed_path, changing the direction of the agents works well here.

3) failed:

This failed error occurs when one of the blocks attached to the agent cannot rotate to its target position, which is to the south in our strategy. As the only parameter to the rotate function is either clockwise or counterclockwise, this error only comes up when the agent cannot rotate in the direction specified. Rotating in the other way, as in rotating clockwise when our initial strategy was to rotate counterclockwise worked in most scenarios.

Future Improvements

If we had more time to develop, we would have implemented an exploration function that makes the agent explore the map before attempting to complete tasks. This would help us save the

coordinates of dispensers and goals, making task completion more efficient later. Additionally, we would introduce agent communication so that each agent can share its local map, giving all the agents a complete and more accurate global map.

With the agents knowing exactly where dispensers and goals are, this would completely remove the need for random movement as they would always be moving in an exact direction either towards a dispenser or towards a goal.

However, we still have an issue where agents attach to multiple blocks due to problems with task submission. To solve this issue, we can assign tasks based on the proximity of dispensers before agents pick up blocks. This ensures that agents will not try to simultaneously submit the same tasks.

Successes in Agent Development

The things our team has done well in strategy and implementation:

The search for dispensers and goals is implemented very well. If the agent can sense both dispensers and goals, it can determine the route and move to the target location.

At the same time, it also prevents two agents from competing for one dispenser. block function. This situation will make both agents unable to work.

In addition, the rotation function is implemented better when agents submit blocks because once the agent reaches the target position, it can immediately rotate and submit. At the same time, it can also respond to rotation failure during submission, which can prevent the agent from always being in the submission stage.

Finally, we have done a decent job in realizing the function of agents avoiding obstacles. We have fully considered the situation that agents need to avoid obstacles when moving. And programmed obstacle avoidance in the case of a block attached, which can ensure the efficient work of agents.

Challenges Faced in Agent Development

On the other hand, the shortcomings of our implementation are:

In implementing random movement of agents, random movement will take a lot of time to sense the dispenser and goal as there is nothing stopping agents from repeatedly visiting the same cells.

We failed to realize that when two blocks submitted blocks at the same time, because the system can only accept one task when executing the removal of belief, one agent would not be able to execute -block attached, which resulted in one of the agents being unable to submit block, and the agent will always attach this block, which will greatly affect the agent's work efficiency.