[DT0171] ARTIFICIAL INTELLIGENCE 2024/2025

PROJECT: The Cooking Chef Problem

Type of submission: Only Project

Student: Tien Dung Nguyen – 301142 – tiendung.nguyen@student.univaq.it

1. States and actions description

As the agent moves, the position changes, the tool is also considered since getting the tool is the first step to complete the task. Therefore, states can be defined as $(x, y, tool\_collected)$ where:

- x, y represent the agent's current position on the grid ($1 \leq x \leq 9, 1 \leq y \leq 4$).
- $tool\_collected$ is the binary flag indicating if the agent has collected the egg beater.

The number of possible states is $(9 \times 4 - 4) \times 2 = 64$.

Possible actions for the agent are moving in four directions and using the gate, which means 5 possible actions. $A = \{up, down, left, right, use\_gate\}$

2. Transition function description $\boldsymbol{P}$: $(\boldsymbol{s'}, \boldsymbol{s}, \boldsymbol{a}) \rightarrow \boldsymbol{P}(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a})$

if a is use_gate:

if s is $(4, 2, collected\_tool)$: $s' = (9, 3, collected\_tool)$

i.e. $P(s'|s, a) = 1$ if $s' = (9, 3, collected\_tool)$ otherwise 0

else if s is $(9, 3, collected\_tool)$: $s' = (4, 2, collected\_tool)$

i.e. $P(s'|s, a) = 1$ if $s' = (4, 2, collected\_tool)$ otherwise 0

else $s' = s$

i.e. $P(s'|s, a) = 1$ if $s' = s$ otherwise 0

else:

try to move agent accordingly

if there is no wall on the way $s' \neq s$

i.e. $P(s'|s, a) = 1$ if s' is in the direction of a otherwise 0

else $s' = s$

i.e. $P(s'|s, a) = 1$ if $s' = s$ otherwise 0

if $new\_position$ in $\{(1, 3), (8, 3)\}$: $tool\_collected = 1$

i.e. $P(s'|s, a) = 1$ if $s'[tool\_collected] = 1$ otherwise 0

The dimension of P is $|S| \times |S| \times |A| = 64 \times 64 \times 5$. The transition function spreadsheet is available here, with each sheet is the action taken, columns represent the previous state s, rows represent the new state s'.

3. Reward function description

The proposed reward function is designed to guide the agent to complete the cooking pudding eggs task, considering states and actions accordingly.

The reward function value depends on many cases as follows:

- **No movement:** Penalize the agent for staying in the same state (e.g., bumping into walls or invalid moves).
- **Using gate:** If the agent uses the gate at position $(4, 2)$, reward it as it leads to the oven, penalizes if use gate at position $(9, 3)$ as it is less optimal and penalizes heavily if the agent tries to use the gate at non-gate locations.
- **Reaching the goal with tool collected:** Rewards the agent for completing the task by cooking the eggs after collecting the tool.
- **Collecting the egg beater:** If the agent has not got the tool, reward it for collecting the tool and penalize otherwise.
- **Moving without the tool:** If the agent is in the left half of the map, penalizes it by the distance to the egg beater at position $(1, 3)$. If the agent is in the right half of the map, penalizes it by the distance to the egg beater at position $(8, 3)$.
- **Moving with the tool:** If the agent is in the left half of the map, penalizes it by the distance to the nearest gate at $(4, 2)$. If the agent is in the right half of the map, penalizes it by the distance to the oven.

This reward function has some key characteristics:

- **Goal-oriented:** Rewards the agent for completing the task efficiently, with high rewards for collecting the tool and cooking the eggs.
- **Distance-based guidance:** Make use of Manhattan to create penalties propotional to the distance from key locations (egg beater, gates, or stove).
- **Error preventing:** Penalizes invalid moves heavily to prevent unuseful actions.
- **Dynamic behavior:** Differentiate between gates to encourage optimal transitions. The reward also depends on the agent's progress (whether the tool is collected) and relative location to the key positions.

The proposed reward function is designed to ensure the agent avoids unnecessary steps, prioritizes collecting the beater before heading to the stove and moves efficiently toward objectives.

4. Policy computation and visualization

In this implementation Q-learning algorithm is used with a ε-greedy policy to balance between exploration and exploitation.

a. **ε-greedy policy:** With a probability of ϵ, a random action is chosen to encourage exploration, otherwise the action with the highest Q-value is selected to exploit the learned knowledge. If multiple actions have the same maximum Q-value, one is randomly chosen to prevent deterministic behavior.
b. **Training loop:** For each training episode
   - **Initialization:** Agent starts at the initial position with the flag $tool\_collected = 0$.
   - **State transition:** Agent selects an action using ε-greedy policy. The agent transits to the next state and the immediate reward is calculated. In case of entering the cell with the egg beater located, update $tool\_collected = 1$.
   - **Q-value update:** The Q-value for the current state-action pair is updated using the Q-learning formula:

$$Q(s,a) = Q(s,a) + \alpha \times [r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

   - **State update:** Agent state is updated, and the loop continues until the agent reaches the stove after collecting the tool.
c. **Policy extraction and visualization**

After training, the optimal policy is derived by selecting the action with the maximum Q-value for the states.

For every state:

- Compute all Q-values for possible actions.
- Select the action with the highest Q-value.

The calculation process takes these parameters:

- Number of loops: 1000
- Learning rate $\alpha = 0.1$
- Discount factor $\gamma = 0.9$
- Random action probability $\epsilon = 0.1$

The computed policy is shown in Figure 1 and Figure 2, with both cases of $tool\_collected = 0$ and $tool\_collected = 1$, where arrows represent the movement action in direction of the arrow, the red G letter represents the action of using the gate.

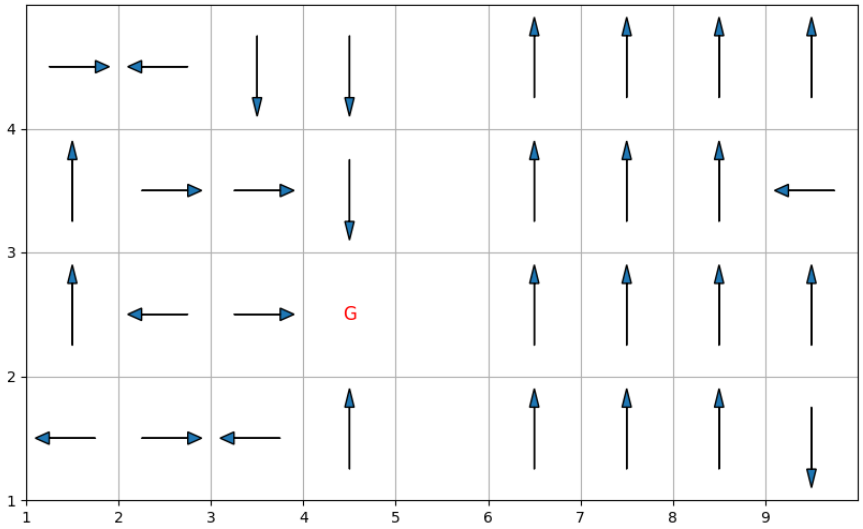

Figure 1. Computed policy in case no tool collected

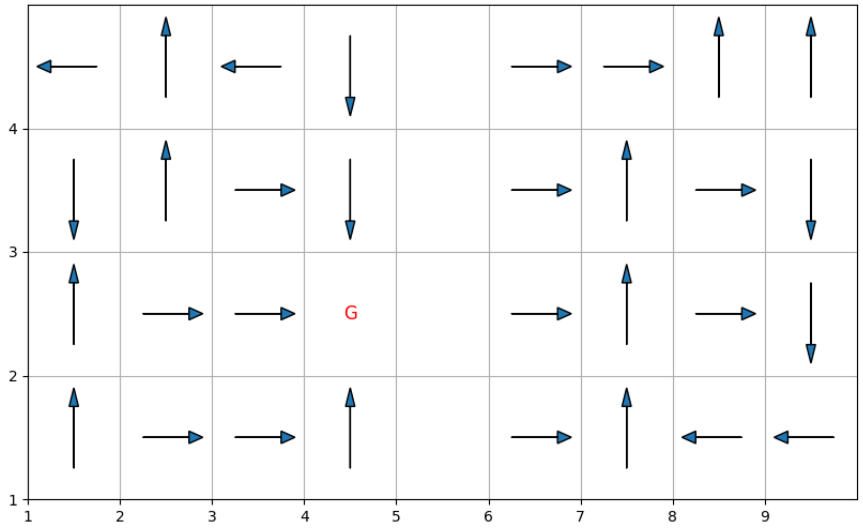

Figure 2. Computed policy in case of tool collected

In case of no tool collected, the agent tends to use moves that take it closer to the egg beater, which yields better reward. Another case is going to the other side of the board using the gate and go to the tool at position (8, 3). For the tool collected scenario, the agent tends to go to the gate and use the gate at (4, 2), then move to the oven at (8, 4), which helps completing the task. Therefore, the proposed reward function helps the agent to learn a quite good policy.

However, the best strategy is sometimes different from the computed policy. For example, at position (1, 1), the agent should go to the right direction. This is due to the penalty given at all cases when the agent cannot collect the tool or go to the oven, the uncalculated action was the best solution although it does not help the agent moving.