AI models for physics exam project: Cooperative Multi-Agent Maze Escape

Valiev Aly, 911861
June 9, 2024

In the scope of this project, two independent agents were trained using the PPO algorithm in the multi-agent environment that encourages the development of cooperative strategies, requiring agents to navigate through the grid, avoid walls, and coordinate to open a door to escape successfully.

I. ENVIRONMENT DESCRIPTION

Both agents are identical and start from two different corners of the maze. In this environment, two agents try to escape together from a maze with barriers. The agents take turns in a step-by-step manner. Additionally, several initialization parameters can slightly change the environment: randomization of the door position, switching the finish cells, making it easier for the agents to 'bump' into each other, and adding an additional wall. These configurations can be combined to create various scenarios.

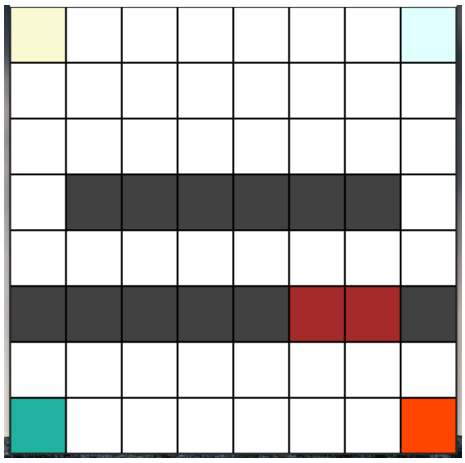

Figure 1. An 8 by 8 grid representing the maze environment. The colors indicate different elements within the maze: dark gray for walls, blue-green for the current position of agent 1, orange for the current position of agent 2, light cyan for the end position of agent 1, light yellow for the end position of agent 2, and brown for a closed door.

The door is considered too heavy for a single agent, so it can be opened only when they push it together. The game ends if both agents achieve their finish cells. Agents are not allowed to move through each other or occupy the same cell.

II. TRAINING ALGORITHM

In this project, the Proximal Policy Optimization (PPO) algorithm to train agents was implemented. The PPO algorithm, a model-free method in reinforcement learning, optimizes the policy by maximizing the expected cumulative reward while ensuring updates are not too big, which helps to stabilize the training. Actor-critic networks estimate the policy and the value function accordingly. Below you can see the pseudocode description of this algorithm from OpenAI's Spinning Up doc.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Figure 2. Pseudocode of PPO

Additionally entropy regularization was added to encourage exploration.

III. RESULTS

The environment was solved for all configurations. First, the easiest configuration with a fixed door position, finish cells on the same columns as starting cells, and no additional walls. Then, gate position randomization and switching the finish cells were added. Finally, the hardest attempt, where an additional wall was included (see Figure 1), was also solved. Let's focus on the last.

We are not providing loss dynamics because in our algorithm, learning targets are non-stationary; they are functions of the actor/critic networks that are being modified. The main metrics in our case are the score (final reward of the agent in a single episode) and the number of moves that were taken to solve the maze. Additionally, we plot the number of the move at which the door was opened.

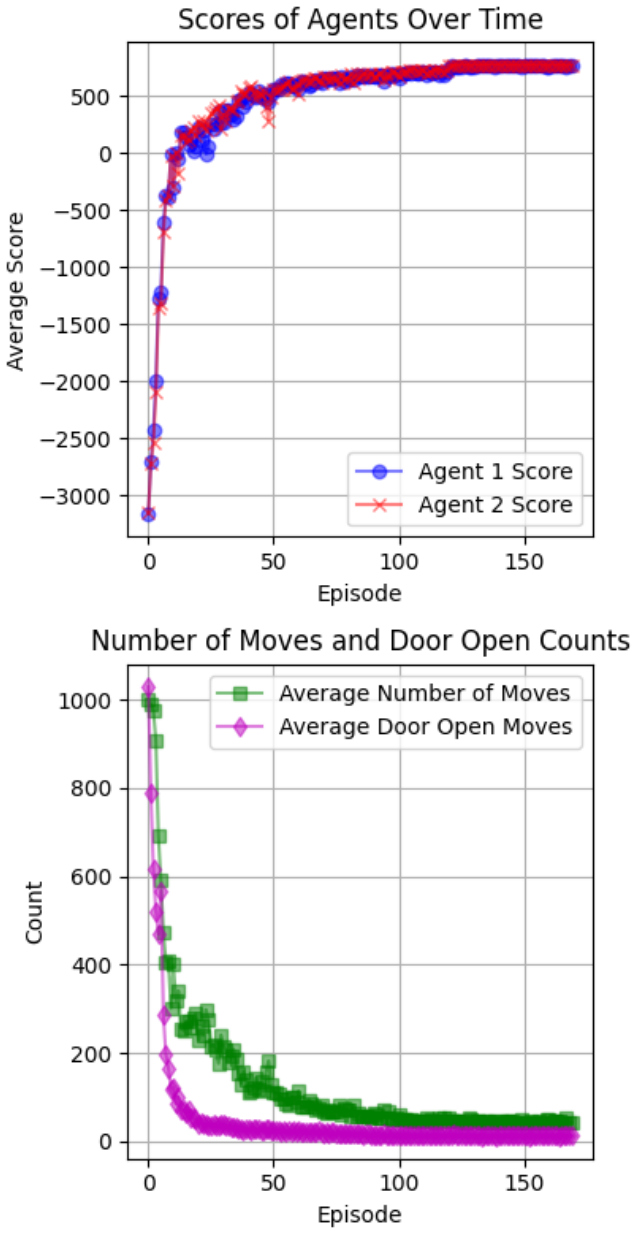

Figure 3. Agent's perfomances

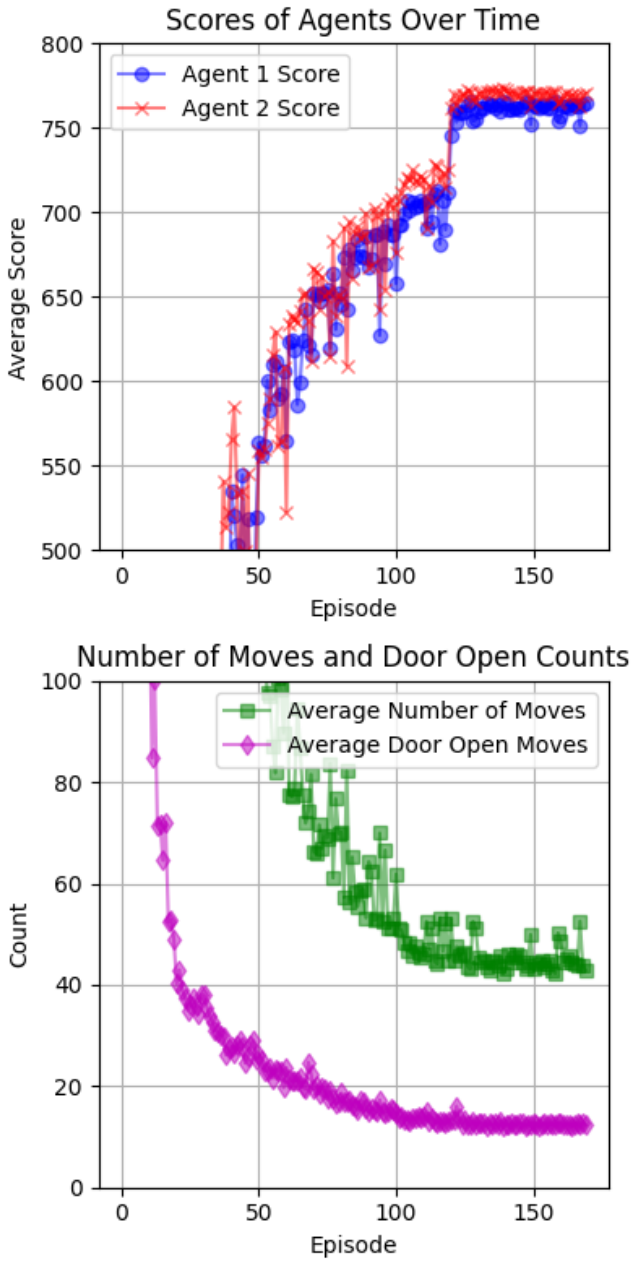

Figure 4. Agent's perfomances (amplified)

Above, we can see how the scores of both agents and the number of moves taken to open the door and finish the maze were changing during training. Agents were able to find the optimal solution.

An interesting increase in the scores can be seen around episode 120, when the entropy regularization coefficient was increased.

All the implementations, code, and videos of agents solving the environment can be found at this project's GitHub repository.