Cairo University

Faculty of Engineering

Computer Engineering Department

CMPS458 Reinforcement Learning Report Template

Team Name/Number: 14
Aya Ayman Mohamed Abdelmegid
Maryam Mostafa Omar Habeb

Ziad Ahmed Mohamed Mahdy Asar

Supervisor: Ayman AboElhassan

October 24, 2025

Deliverables

This section summarizes the key deliverables submitted as part of the project, including the source code repository, recorded demonstration video, and a representative visualization of the Grid Maze environment used in the experiments.

- **Source Code Repository:** The full implementation of the Grid Maze Environment and Policy Iteration algorithm is available on GitHub.

  https://github.com/ayahayman/Reinforcement-Learning-Policy-Iteration-Grid-Maze-

- **Video Demonstration:** A recorded video showcasing the trained agent navigating through the maze environment using the derived optimal policy is available at:

  https://drive.google.com/drive/u/2/folders/1T_Yk1RA41qmMLLMUXG5DO6mw6OIgvGdc

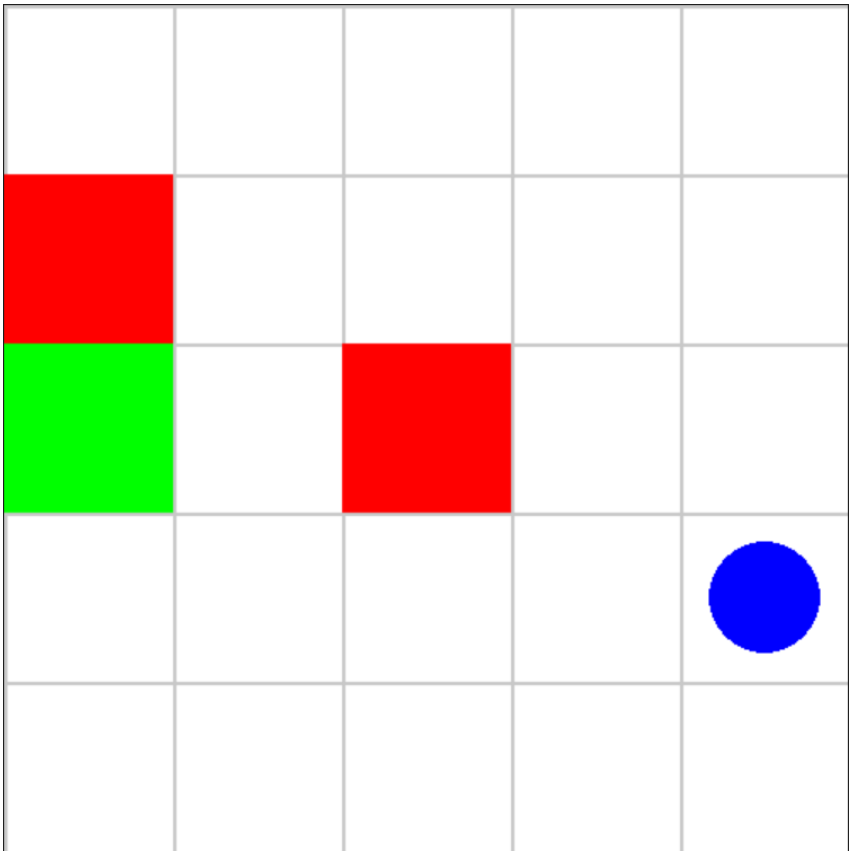

Figure 1: A visual sample of the Grid Maze environment used for Policy Iteration training and evaluation.

Discussion

0.1 Experiments

This section discusses the conducted experiments and their outcomes, focusing on how the results change when varying key parameters of the Policy Iteration algorithm and the environment configuration.

0.1.1 Overview of Experiments

A series of experiments were conducted on Grid Maze environments of various sizes to observe the agent's learning behavior and policy convergence. Each experiment involved training the agent using Policy Iteration on a discrete grid world where the agent receives:

- A **positive reward** upon reaching the goal cell,
- A **negative reward** when entering a bad cell, and
- A small **step penalty** to encourage shorter paths.

The goal of these experiments was to analyze how the algorithm's performance and the resulting policy are affected by the discount factor (γ), reward configuration, and convergence threshold (θ).

0.1.2 Effect of Discount Factor (γ)

The discount factor γ controls how much the agent values future rewards compared to immediate ones.

- **Low γ (e.g., 0.3–0.5):** The agent becomes highly **greedy**, prioritizing immediate rewards. It may take risky shortcuts or move closer to bad cells if the negative values are small, since it heavily discounts future penalties. This behavior leads to faster convergence but less optimal long-term planning.
- **High γ (e.g., 0.9–0.99):** The agent values long-term rewards more, resulting in **safer and more strategic paths**. However, this also makes convergence slower, as future rewards have more influence and require more policy evaluation updates.

0.1.3 Effect of Convergence Threshold (θ)

The threshold θ determines the stopping condition for value updates. A smaller θ allows for more accurate estimation of the value function but increases computation time.

- **Large θ (e.g., 10^{-3}):** Faster convergence but may result in a slightly suboptimal policy.
- **Small θ (e.g., 10^{-6}):** More precise value estimation and stable policy, at the cost of additional iterations.

0.1.4 Effect of Grid Size

When increasing the grid size (e.g., from 5×5 to 10×10), the number of possible states grows quadratically. This significantly increases computation time, as Policy Iteration must evaluate and improve policies for all states.

- **Smaller grids (5x5):** Converged quickly, typically within 4–5 iterations.
- **Larger grids (10x10):** Required more iterations and exhibited slower convergence, though the same principles applied.

0.1.5 Summary of Observations

- Low γ values produce **greedy agents** that may take riskier actions for faster rewards.
- High γ values lead to more **strategic and cautious behavior**, with longer planning horizons.
- Smaller θ values increase accuracy but also computational cost.
- Increasing the grid size exponentially increases the state-space, making convergence slower but the learned policy more general.

0.2 Question Answers

0.2.1 1. What is the state-space size of the 5x5 Grid Maze problem?

The **state-space size** represents the total number of possible states in which the agent can exist. In a 5×5 grid, each cell corresponds to one possible state:

$$\text{State-space size} = 5 \times 5 = 25$$

Thus, there are **25 distinct states** in total.

However, if we consider all possible configurations of the environment — such as different placements of the start cell, target cell, and two bad cells — the total combinations would be:

$$25 \times 24 \times 23 \times 22$$

Training on all such combinations would be computationally expensive and time-consuming.

0.2.2 2. How to optimize policy iteration for the Grid Maze problem?

Policy iteration can be optimized in several ways:

- **Early Convergence Check:** Stop the iterations once the policy becomes stable to prevent unnecessary computation.
- **Value Function Thresholding:** Detect convergence early by monitoring the change in the value function and stopping when it falls below a small predefined threshold (θ).

0.2.3 3. How many iterations did it take to converge on a stable policy for the 5x5 maze?

For a 5×5 grid maze, the algorithm typically **converges within 4 to 5 iterations** on average.

0.2.4 4. How does policy iteration behave with multiple goal cells?

When multiple goal cells exist, the agent typically **favors the goal cell that yields the highest expected return**. In most cases, this will be the goal cell that is **closer to the agent**, since it requires fewer steps and thus less discounting of rewards.

However, if different goal cells have varying reward values, the agent may instead favor a more distant goal if it provides a higher total return.

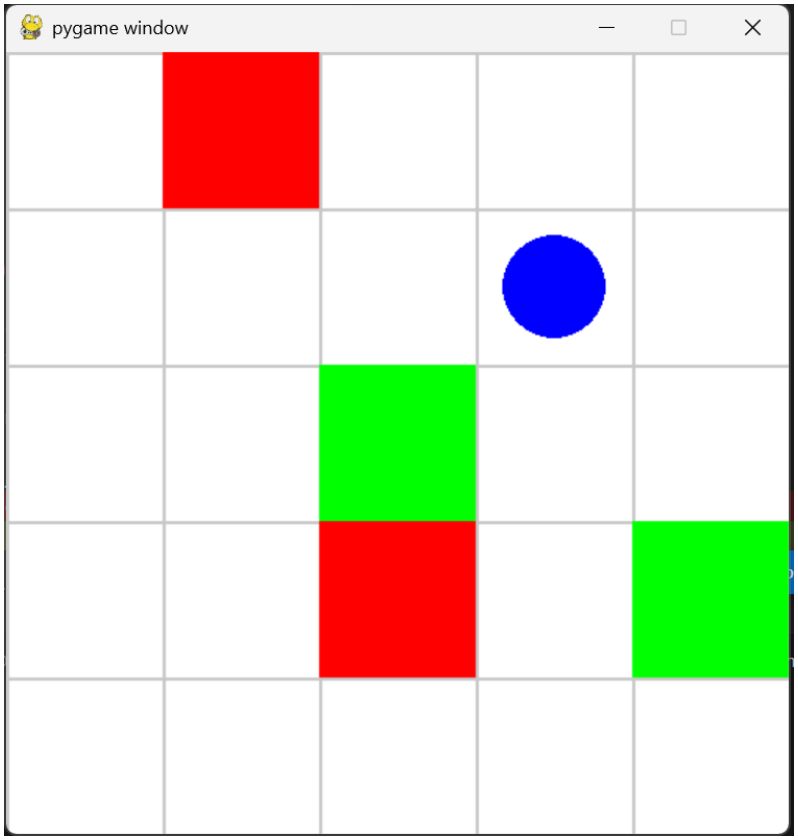

Figure 2: A sample of the Grid Maze environment with two goal cells.

0.2.5 5. Can policy iteration work on a 10x10 maze? Explain why.

If the agent is trained on a 5×5 maze and tested on a 10×10 maze, it will **not generalize effectively**. The learned policy and value function are specific to the smaller grid's state-space. The larger maze introduces **new, unseen states and spatial relationships**, making the existing policy invalid or suboptimal.

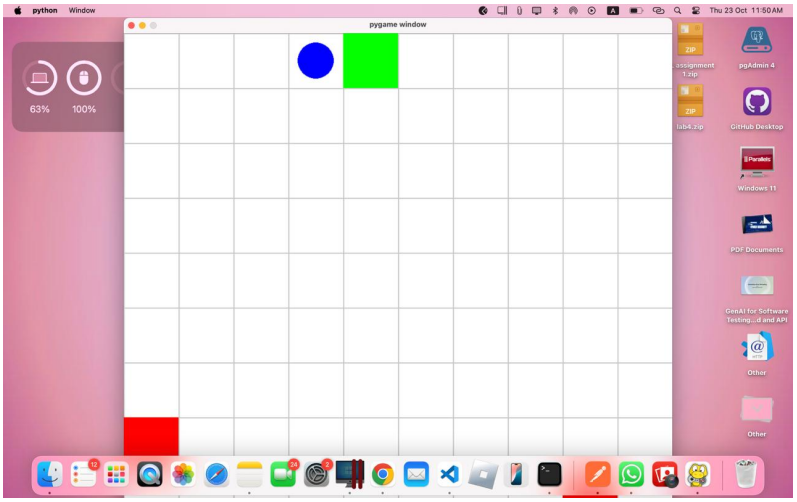

Figure 3: A sample of a 10x10 Grid Maze environment.

0.2.6 6. Can policy iteration work on a continuous-space maze? Explain why.

Not effectively. Policy iteration is designed for **discrete and finite state spaces**. In a continuous environment, the summations in the Bellman equations become integrals, and the agent's possible positions become infinite. This makes it infeasible to compute or store a value function for all possible states, preventing convergence.

0.2.7 7. Can policy iteration work with moving bad cells (like Pac-Man ghosts)? Explain why.

Policy iteration cannot effectively handle moving bad cells unless their movement is **deterministic or follows a known pattern**. Standard policy iteration assumes a **stationary environment**, where transition probabilities between states remain fixed.

If bad cells move unpredictably or randomly, the environment becomes **non-stationary and time-dependent**, violating the Markov assumption. As a result, the learned policy would quickly become invalid, since the transition dynamics change over time.