Cairo University

Faculty of Engineering

Computer Engineering Department

CMPS458 Reinforcement Learning- Assignment 1

Team Name/Number: Team 1
First member Name: Mariam Mahrous 1210301
Second Member name: Menna Salah 1210032

Third member name: Farida Ahmed 1210276

Supervisor: Ayman AboElhassan

October 23, 2025

Deliverables

Repo link: https://github.com/Mennasalah140/Reinforcement-Learning

Video record link:
https://drive.google.com/drive/folders/1qKALzJyihPuTI0yaEyy-LIOcd4qJNgRN?usp=sharing

Discussion

0.1 Question Answers

Q1 — What is the state-space size of the 5×5 Grid Maze problem?

There are three possible interpretations of the state space depending on how we model the environment:

Table 1: State Space Interpretations

Interpretation	State Definition	State Space Size	Notes
Policy Iteration (correct for DP)	Agent position only `(Sx, Sy)`	$5 \times 5 = 25$	Used for Dynamic Programming (fixed MDP)
Full environment observation	Agent, Goal, X1, X2 coordinates	$5^8 = 390,625$	Used only in Gym observation, not DP
All distinct cell placements	S, G, X1, X2 in unique cells	$25 \times 24 \times 23 \times 22 = 303,600$	Represents layout permutations

Q2 — How to optimize the policy iteration for the Grid Maze?

Policy Iteration can be optimized on two levels:

A) Local Optimization (inside DP) — Faster, same results

These optimizations reduce computation time without changing the final policy:

- **Stop policy evaluation early** using a convergence threshold
- **Solve For fixed Maze instance** this reduces the state space from the full $390,625$-state observation space to the 25-state DP model

B) Global Optimization (beyond DP) — Better scalability

For larger or dynamic environments, the best "optimization" is to **replace DP entirely**:
Summary:

- Local optimization = **faster Policy Iteration**
- Global optimization = **use a better RL approach for bigger mazes**

Table 2: Global Optimization (Replacing DP)

Replace DP With	Why it scales better
Q-Learning / SARSA	No need to evaluate all states or know transition probabilities
Monte-Carlo / TD methods	More efficient than DP for bigger problems

Q3 — How many iterations did it take to converge?

This depends entirely on the maze generated. We have tried several random mazes, and it typically takes 3 to 7 iterations for the policy to converge.

Q4 — How does policy iteration behave with multiple goal cells?

Policy Iteration will:

- Treat all goal states as terminal
- Generate **multiple value gradients**
- Drive the agent toward the **nearest goal** (shortest expected path)

Example: If goals are at `(0,4)` and `(4,4)`:

- Agent at `(2,4)` will move **up**
- Agent at `(4,2)` will move **right**

The optimal policy adapts to the closest rewarding terminal state.

Q5 — Can Policy Iteration work on a 10×10 maze?

Yes, because the environment is still **finite and discrete**. However, it becomes **computationally slower**, since Policy Iteration must evaluate **all states in every iteration**:

- $5 \times 5 \rightarrow 25$ states
- $10 \times 10 \rightarrow 100$ states ($4\times$ more than 5×5)

It works, but scales poorly as the grid grows.

Q6 — Can Policy Iteration work on a continuous-space maze?

No, not in its classical tabular form. Policy Iteration requires:

- A **finite enumerable state space**
- A **known transition model**

Continuous environments have **infinite states**, so DP cannot run. Instead, we must use **function approximation or Deep RL.**

Q7 — Can Policy Iteration work with moving bad cells (like ghosts)?

No. Moving enemies make the environment **non-stationary**, which violates Policy Iteration assumptions. DP requires a **fixed transition model** that does not change over time. For dynamic environments, use **online RL methods** such as:

- Q-Learning
- SARSA