1.

Cairo University
Faculty of
Engineering

Computer Engineering Department

# CMPS458 Reinforcement Learning Lab-1 Report

Team Number: 4
Mostafa Osama
Maged Amgad

*Supervisor*: Ayman AboElhassan

October 22, 2025

# Deliverables

Repo link: https://github.com/MagedWadi/RL-ASS-1
Video record link: https://github.com/MagedWadi/RL-ASS-1
are found in the videos folder


Figure1:Relation between Agent and Environment

# Discussion

## 0.1 Experiments

This section discusses the experiments performed on the Grid Maze environment and the observed outcomes. Each case studies how different parameters influence convergence, policy behavior, and computational performance.

### Case 1 — Default Parameters

- Grid size: $5 \times 5$
- Discount factor: $\gamma = 0.95$
- Goal: bottom-right corner (state 24)
- Two bad cells: random locations
- Convergence threshold: $\theta = 1 \times 10^{-6}$

**Result:** The algorithm converged after 6 iterations. The value map showed higher values near the goal and lower values near bad cells. The optimal policy pointed toward the goal from most states while effectively avoiding bad cells.

### Case 2 — Changing Discount Factor ($\gamma$)

We tested $\gamma \in \{0.5, 0.8, 0.95, 0.99\}$.

| $\gamma$ | Convergence Iterations | Behavior |
|---|---|---|
| 0.5 | 3 | Prefers immediate rewards, ignores long paths |
| 0.8 | 5 | More balanced decisions |
| 0.95 | 6 | Stable and goal-oriented |
| 0.99 | 8 | Slower convergence, smoother value distribution |

**Observation:** Higher $\gamma$ values make the agent plan further ahead, improving goal-seeking behavior but increasing computation time.

### Case 3 — Multiple Goals

When two goal cells were placed (top-right and bottom-left corners), the grid split into two distinct attraction zones. States closer to each goal directed the agent toward that goal.

**Result:** Convergence occurred after 5 iterations.

**Insight:** Policy iteration naturally handles multiple terminal states by optimizing for the nearest high-value region.

### Case 4 — Larger Grid ($10 \times 10$)

When the grid size was increased to $10 \times 10$ (100 states):

**Result:** Convergence required 9–11 iterations. The algorithm remained stable but required more computation time due to the larger state space.

**Observation:** As the number of states increases, both policy evaluation and improvement steps become more computationally intensive. However, the method continues to converge reliably in finite discrete environments.

## 0.2 Question Answers

### Question 1: What is the state-space size of the 5x5 Grid Maze problem?

In a 5×5 grid, each cell represents a possible agent position.

**Total states = 5 × 5 = 25.**

### Question 2: How to optimize the policy iteration for the Grid Maze problem?

Policy iteration can be accelerated through several optimization techniques:

- **Vectorization:** Use NumPy arrays instead of nested loops to update all states at once.
- **Partial evaluation:** Limit value updates to a few sweeps (e.g., 3–5) before improving the policy.
- **Transition caching:** Precompute and store transition outcomes for static environments.
- **Hybrid method:** Combine evaluation and improvement steps (as in Value Iteration) for faster convergence.

### Question 3: How many iterations did it take to converge on a stable policy for the 5x5 maze?

With parameters like `reward(goal)=+10`, `reward(bad)=-10`, `step cost=-1`, and `γ=0.95`, policy iteration typically stabilizes within **4–6 iterations**.

**Example:** Converged after 6 iterations.

### Question 4:Explain, with an example, how policy iteration behaves with multiple goal cells.

When multiple goals exist, each functions as a terminal state with a positive reward.
**Example:** For goals at (4,4) and (0,4), both with +10 reward, the algorithm:

- Assigns high state values around both goals.
- Learns two regions of attraction—one near each goal.
- Guides the agent to the closest goal based on its start location.
  Ultimately, the policy partitions the grid into zones leading toward different goals.

### Question 5: Can policy iteration work on a 10x10 maze? Explain why.

Yes. The algorithm still works because the environment is finite and discrete.

**State count:** $10 \times 10 = 100$

**State–action pairs:** $100 \times 4 = 400$

However, computation grows roughly with **$O(n^2)$**, so larger grids take more time.

Optimizations like vectorization and truncated evaluation can offset this cost.

### Question 6: Can policy iteration work on a continuous-space maze? Explain why.

Not directly. The method relies on enumerating all states and policies:

- $V(s)$ and $\pi(s)$ must be stored for every state.
  In continuous domains, infinite states make this impossible.
  To address this, **function approximation** is used, forming the basis of:
- **Approximate Policy Iteration**
- **Actor–Critic algorithms**
  These techniques belong to **Deep Reinforcement Learning**.

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

Not in its standard form. Policy iteration assumes **stationary transition probabilities** $P(s' \mid s,a)$.
When obstacles move, the environment becomes **non-stationary**.
To adapt, one must:

- Include dynamic elements (like ghost positions) as part of the state.
  This dramatically increases state complexity, so **model-free methods** (e.g., Q-learning, DQN) are preferred for such scenarios.