Project 3: Markov Decision Process Navigation

Group - 34

Name: Satya Tej Kammili

Student ID: 11911161

1. Problem Statement

This project involves designing an intelligent navigation system using a Markov Decision Process (MDP) within a 5×8 grid world. The grid contains hazard states that must be avoided and a goal state that the robot must reach. Movement is stochastic due to environmental noise, and allowable transitions are constrained by directional rules provided in Tutorial 5. The objective is to determine:

1. **R1:** The first live-in reward value r∈[−20,0]r \in [-20, 0]r∈[−20,0] that produces a valid optimal policy reaching the goal from all start states.
2. **R2:** The modified optimal policy when an obstacle is added at position (3,4), and an explanation of how and why the policy changes.
3. **R3:** A conceptual extension to prevent collisions between multiple robots operating simultaneously in the grid.

The task includes full MDP formulation, implementation of value iteration, extraction of optimal policies (P1 and P2), and a detailed written explanation of results.

2. Environment Description

The environment is a discrete grid with the following key components:

1. **Goal state:** (1,3) with reward **+1000**
2. **Hazards:** (3,1) and (4,1) with reward **–1000**
3. **Other states:** Live-in reward r∈[−20,0]r \in [-20, 0]r∈[−20,0]
4. **Movement model:** 8 directions

   ** N, NE, E, SE, S, SW, W, NW **

5. **Noise:** 0.1 (intended move succeeds with probability 0.9)

The permitted directions for any intended direction follow the rules in **Tutorial 5**, where directions at **90° or 180°** from the intended direction are blocked.

Boundaries force the robot to stay in place when a movement would result in leaving the grid.

3. Transition Model

The MDP transition probabilities reflect both noise and movement restrictions:

1. Intended direction aaa: probability **0.9**
2. Remaining permitted directions share the **0.1** noise
3. Blocked or out-of-bounds moves result in **remaining in the same state**
4. Obstacles (for R2) also force a stay-in-place transition

The transition model is used inside the Bellman update during value iteration.

4. Value Iteration Method

Value iteration applies the Bellman equation:

U(s)=r(s)+γamax s′∑ T(s,a,s′)U(s′)

Where:

1. γ=0.99\gamma = 0.99γ=0.99 (discount factor)
2. r(s)r(s)r(s) is the state reward
3. Transition probabilities T(s,a,s′)T(s,a,s')T(s,a,s′) come from the movement model

Iterations continue until convergence (Δ<0.001\Delta < 0.001Δ<0.001), ensuring stable utility values across the grid.

5. R1 — Live-in Reward Search & Optimal Policy P1

All values of r∈[−20,0]r \in [-20, 0]r∈[−20,0] were tested using value iteration.
Each resulting policy was validated by simulating movement from all 7 start states:

Start states:
(1,1), (2,1), (2,2), (3,2), (4,2), (5,2), (5,3)

First valid live-in reward:

r = –20

This was the **first** reward in the range that successfully produced a policy allowing the robot to reach the goal from **every** start state.

Policy P1 (with arrows):

```
↘ ↘ ↘ ↓ ↓ ↙ ↙ ↙
H → ↘ ↓ ↙ ↙ ↙ ↙
H → ↘ ↙ ↙ ↙ ↙ ↙
↓ ↘ ↓ ↙ ↙ ← ← ←
→ → G ← ← ← ← ↖
```

6. R2 — Obstacle at (3,4) and Updated Policy P2

An obstacle was added at **(3,4)**, and value iteration was recomputed using the same reward r=–20r = –20r=–20.

The new optimal policy P2 is:

↘ ↘ ↘ ↓ ↙ ↓ ↙ ↙

H → ↓ ↙ ↓ ↙ ↙ ↙

H → ↘ ↙ ↙ ↙ ↙ ↙

↓ ↘ ↓ ↙ ← ← ← ←

→ → G ← ← ← ← ↖

Comparison Between P1 and P2

The obstacle at (3,4) introduces a local detour in the surrounding area:

1. Some diagonal movements become downward or leftward movement
2. The robot avoids the obstacle by taking alternative safe routes
3. Despite changes in arrow patterns, **reachability remains unchanged**

Conclusion for R2

Both P1 and P2 are successful policies, but **P2 locally reroutes** around the obstacle.
Global structure remains intact; only paths near the obstacle change.

7. R3 — Extending the MDP for Multi-Robot Collision Avoidance

In multi-robot navigation, each robot's motion depends on the predicted positions of other robots. To prevent collisions, the transition model is adjusted to incorporate **occupancy probability**:

T′(s,a,s′)=T(s,a,s′)×(1−Poccupied (s′))

Where:

1. Poccupied(s′)P_{\text{occupied}}(s')Poccupied (s′) is the probability that another robot will occupy state s′

Cells with high collision risk have lower transition probability, making the robot avoid them.

Because joint multi-robot MDPs grow exponentially with robot count, practical systems use:

1. Factored MDPs
2. Local interaction zones
3. Priority / yielding rules
4. Motion prediction for neighboring robots

This ensures safe navigation without excessive computational cost.

8. Conclusion

The MDP-based navigation system successfully computed optimal policies for both the original and obstacle-modified environments. The first valid live-in reward was **r = –20**, which produced a fully functional policy guiding the robot safely to the goal. Introducing an obstacle affected local movement but did not prevent convergence to the goal. The framework also extends naturally to multi-robot settings by incorporating occupancy probabilities into the transition model.