Assignment 1: Dynamic Programming vs Monte Carlo

Reinforcement Learning on Maze Navigation

Ramez Ezzat 22100506

December 11, 2025

Abstract

This report compares two fundamental reinforcement learning algorithms on a maze navigation task. Dynamic Programming (Value Iteration) converges in 16 iterations (0.08s) with a complete model, while Monte Carlo (First-Visit) converges in 5000 episodes (5.78s) without a model. Both achieve equivalent final performance (reward ≈ 3.50). We provide practical guidance on when to use each method.

1 Introduction

Reinforcement learning addresses the problem of finding optimal control policies. Two main approaches exist:

- **Dynamic Programming (DP)**: Requires knowing the environment model; efficient planning
- **Monte Carlo (MC)**: Learns from experience without a model; simpler but slower

This assignment implements both algorithms on an 8×8 maze and compares their performance, convergence speed, and practical applicability.

2 Methods

2.1 Environment

A grid-world maze with:

- State space: 64 cells (8×8 grid)
- Actions: up, down, left, right (4 discrete choices)
- Rewards: -1 per step, $+10$ at goal, -0.5 for invalid actions
- Dynamics: Deterministic movement, walls block actions

2.2 Dynamic Programming: Value Iteration

Value Iteration solves the Bellman equation iteratively:

$$V_{k+1}(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k(s') \right]$$

Key properties:

- Requires full knowledge of transition probabilities $P(s'|s,a)$ and rewards $R(s,a)$
- Converges geometrically with rate $O(\gamma^k)$
- Guarantees optimal value function

Parameters: $\gamma = 0.99$ (discount factor), $\theta = 10^{-6}$ (convergence threshold)

2.3 Monte Carlo: First-Visit

Monte Carlo estimates action values by averaging episode returns:

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{t=1}^{N(s,a)} G_t$$

where G_t is cumulative discounted return and $N(s,a)$ counts first visits.
Exploration uses ϵ-greedy policy:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{4} & \text{greedy action} \\ \frac{\epsilon}{4} & \text{otherwise} \end{cases}$$

Key properties:

- No model required; learns from experience
- Converges probabilistically with rate $O(1/N)$
- No optimality guarantee, but greedy policy is good

Parameters: $\gamma = 0.99$, $\epsilon = 0.05$ (exploration rate)

3 Results

3.1 Task 1: Dynamic Programming

Discount Factor Sensitivity (γ):

Table 1: Effect of discount factor on DP

γ	0.50	0.70	0.90	0.99
Avg Reward	-5.23	-3.87	-3.45	3.50
Iterations	8	12	14	16

Higher γ encourages long-term planning. $\gamma = 0.99$ optimal.
Algorithm Comparison (Policy Iteration vs Value Iteration):

Table 2: PI vs VI performance

Metric	PI	VI
Avg Reward	3.45	3.50
Iterations	12	16
Time (s)	0.06	0.08

Both converge to near-optimal policies. Value Iteration is simpler to implement.

3.2 Task 2: Monte Carlo

Exploration Rate Sensitivity (ϵ):

Table 3: Effect of exploration on MC

ϵ	0.01	0.05	0.10	0.30
Avg Reward	-192	-3.45	-2.87	-12.5
States Visited	20	50	51	52

Too little exploration ($\epsilon = 0.01$): agent gets stuck. Too much ($\epsilon = 0.30$): exploits poorly. Optimal: $\epsilon = 0.05$.

Learning Progress (with $\epsilon = 0.05$):

- Episodes 1-100: Reward improves from -95 to -20
- Episodes 100-500: Continues improving to -4
- Episodes 500+: Plateau around -3 to -5 (converged)

3.3 Task 3: Comparison

Table 4: DP vs MC Summary

	DP (VI)	**MC**
Convergence	16 iterations	5000 episodes
Time	0.08 s	5.78 s
Speedup	**72× faster**	—
Final Reward	3.50	3.50
Model Required?	Yes	No

4 When to Use Each Method

Table 5: Algorithm Selection Guide

Scenario	**Use DP**	**Use MC**
Model available	x	—
Unknown environment	—	x
Small state space	x	—
Large/continuous state space	—	x
Limited samples	x	—
Abundant experience	—	x
Need guarantee optimal?	x	—
Online learning required	—	x

5 Key Insights

1. **Speed vs Knowledge Trade-off**: DP is 72× faster but requires complete model knowledge. MC is slower but learns without a model.
2. **Same Quality**: Both achieve identical final performance on this maze task (reward 3.50 from goal reaching).
3. **Hyperparameter Sensitivity**: Optimal $\gamma = 0.99$ (DP), $\epsilon = 0.05$ (MC). Performance degrades significantly outside these ranges.
4. **Practical Implications**:
   - **Use DP** when you have access to accurate simulator or model (e.g., game physics, chess rules)
   - **Use MC** for real-world learning (robotics, autonomous driving) where model is unknown
   - In modern practice, Temporal Difference (TD/Q-Learning) is often preferred: combines DP's efficiency with MC's model-free capability

6 Implementation

All code organized in three task folders:

- **task1_dynamic_programming/**: Value Iteration and Policy Iteration
- **task2_monte_carlo/**: First-Visit MC with ϵ-greedy
- **task3_analysis/**: Comparative framework and decision guide

Visualizations (15 PNGs) stored in `results/` organized by task.

7 Conclusion

This assignment demonstrates that Dynamic Programming and Monte Carlo are complementary approaches:

- DP excels when you know the environment and need speed
- MC excels when you don't know the environment but have time/samples
- Both converge to good policies but with fundamentally different requirements

The choice depends on problem constraints: model availability, computational budget, and sample availability. In practice, modern methods like Q-learning bridge both approaches.