Reinforcement Learning for Maze Solving: A Comparative Study

Sai Prasad Reddy Kukudala (CS 5033) Sandeep Ramesh (CS 5033)

Abstract

This project uses reinforcement learning approaches to solve the Maze solving problem. This project explores the application of reinforcement learning (RL) algorithms to solve maze problems. We compare the performance of Q-learning and Monte Carlo methods implemented on a 10x10 grid, with Sarsa and Dynamic Programming implemented on an 6x6 grid. A novelty introduced in this work is the incorporation of a curiosity factor into the Q-learning and Monte Carlo algorithms. Our results show how these algorithms learn to navigate through mazes efficiently, with the curiosity factor enhancing exploration.

1. Project Domain

1.1. Introduction to Maze Solving

Maze solving is a well-known problem in artificial intelligence and computer science, with applications in different fields like robotics, self-navigation etc. The goal is for an agent to move through a maze and reach a proposed goal while making the shortest and most efficient decisions that collect maximum rewards with fewer penalties. Traditional algorithms like BFS and DFS explore the maze in a predefined manner, leading to unnecessary computations and inefficiencies. RL-based agents can improve through trial and error, learning from previous attempts to improve their decision-making process.

1.2. Environment Setup

We use two different grid sizes for our experiments: a 10x10 grid for Q-learning and Monte Carlo, and an 6x6 grid for Sarsa and Dynamic Programming. The environment is modeled using Gymnasium, allowing us to define states, actions, and rewards.

1.3. State and Action Space

The state of the agent is defined by its current position in the maze, and the action space consists of moving up, down, left, or right(4 directions). Rewards are given for reaching the goal and penalties for hitting walls or taking unnecessary steps. Rewards are assigned based on reaching the goal (+10), making valid moves (-0.05), hitting walls (-1), or revisiting states (-0.1).

2. Hypotheses

1. Q-learning vs Monte Carlo: Q-learning will outperform Monte Carlo in terms of convergence speed and optimal policy discovery on the 10x10 grid.

2. Sarsa vs Dynamic Programming: Sarsa will learn faster than Dynamic Programming on the 6x6 grid due to its ability to handle continuous exploration.

3. Curiosity Factor Impact: The introduction of a curiosity factor will improve exploration efficiency in both Q-learning and Monte Carlo methods.

3. Experiments

3.1. Hypotheses-1 (Q-learning vs Monte Carlo)

Sai Prasad experiments focuses on exploring the impact of the curiosity-factor within the Q-learning algorithm implementation. The curiosity-factor, denoted as λ in my implementation, controls the weight of an intrinsic reward that encourages exploration. Specifically, the intrinsic reward is calculated as $\lambda/\sqrt{N(s)+1}$, where $N(s)$ is the number of times state s has been visited. This reward is added to the environment's reward signal to guide the agent towards less explored states. We implemented Q-learning and Monte Carlo with and without the curiosity factor on the 10x10 grid. The algorithms were trained over 20,000 episodes, with an epsilon-greedy policy used for exploration.

Figure 3 illustrates the cumulative rewards over episodes for two reinforcement learning algorithms Q-Learning and Monte Carlo applied to a maze-solving task. The algorithms are compared with and without curiosity-driven exploration mechanisms. Initially, all four configurations (Q-Learning with curiosity, Q-Learning without curiosity, Monte Carlo with curiosity, and Monte Carlo without curiosity) exhibit steeply negative cumulative rewards, indicating poor performance as the agents explore the maze and learn optimal policies. However, as episodes progress, the cumulative rewards improve across all configurations, reflecting the agents gradual learning of effective strategies.

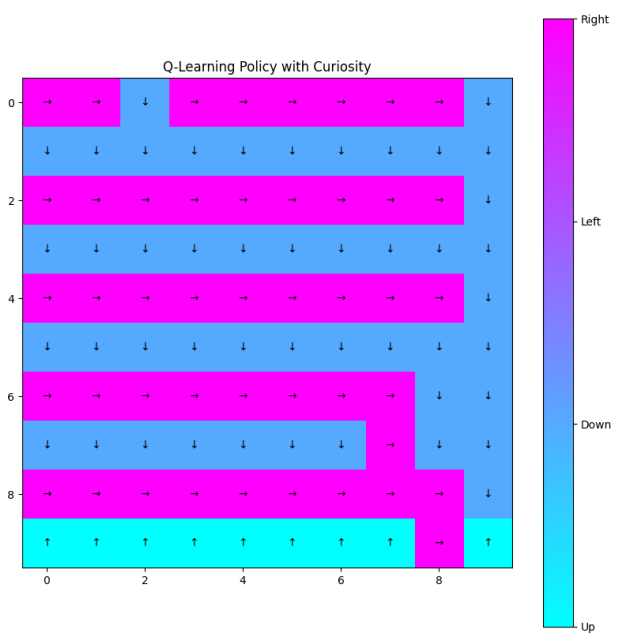

Figure 1. Optimal Policy Learned by Q-Learning with Curiosity.

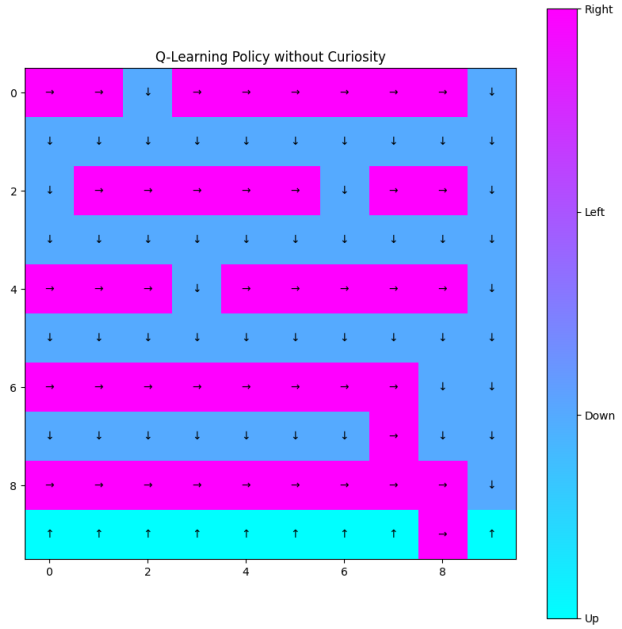

Figure 2. Optimal Policy Learned by Monte Carlo with Curiosity.

Curiosity-driven exploration significantly increases performance for both algorithms. The lines representing "with curiosity" (blue for Q-Learning and green for Monte Carlo) consistently outperform their "without curiosity" counterparts (orange for Q-Learning and red for Monte Carlo). This improvement suggests that curiosity helps agents explore the environment more effectively, allowing them to discover better solutions faster. Moreover, Q-Learning with curiosity demonstrates the fastest convergence and highest cumulative rewards, indicating its superior ability to balance exploration and exploitation compared to Monte Carlo.

In comparing the two algorithms, Q-Learning generally outperforms Monte Carlo in this maze-solving task. This is evident from its higher cumulative rewards throughout most episodes. The difference likely arises from Q-Learning's ability to update state-action values incrementally after each step.

Figure 4 presents the number of steps taken to reach the goal over episodes for Q-Learning and Monte Carlo reinforcement learning (RL) algorithms, both with and without curiosity. Initially, all methods take a significantly high number of steps, often exceeding 2000–3000, as the agents explore the maze with little knowledge of the optimal paths. However, as learning progresses, the number of steps decreases rapidly, showing the agents' improvement in solving the maze efficiently.

Among the approaches, Q-Learning with Curiosity and Monte Carlo with Curiosity demonstrate a slightly faster reduction in steps compared to their non-curiosity algorithms.

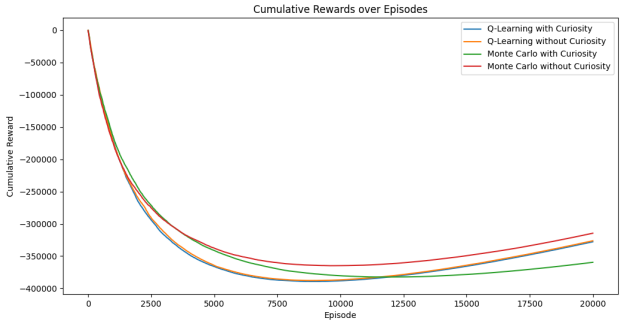

Figure 3. Cumulative Rewards Comparison across Algorithms.

This suggests that curiosity-driven exploration helps agents discover optimal paths more efficiently. Monte Carlo without Curiosity (red) appears to have the slowest convergence, as it exhibits higher fluctuations over a longer duration. By around 10,000 episodes, all methods stabilize with minimal variation, indicating near-optimal policy convergence.

Overall, Q-Learning with Curiosity emerges as the most effective approach, reaching optimal solutions faster. Monte Carlo without Curiosity performs the worst, requiring more steps even in later episodes.

3.2. Hypotheses-2 (Sarsa and Dynamic Programming)

Sandeep implemented Sarsa and Dynamic Programming on the 6x6 grid. It is determined that many reinforcement learning algorithms like SARSA and Dynamic programming can help the agent find the optimal path much faster, which will

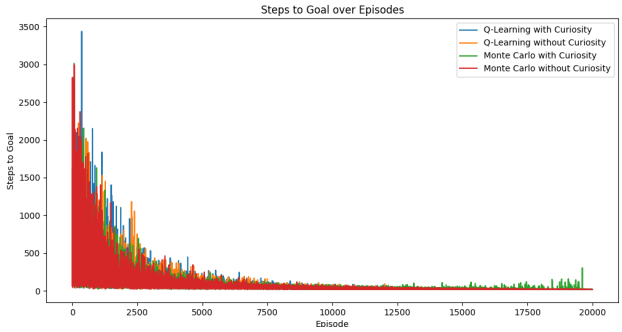

Figure 4. Steps to Goal over Episodes for Q-learning and Monte Carlo (with/without Curiosity)

help people or machines trapped in maze to find the faster way out. In Dynamic programming, value iteration helps to determine the best optimal path in the fastest time such as when we find V*. In SARSA, we can make the agent find the optimal path much quicker if we can increase the learning rate besides increasing the amount of episodes. SARSA was evaluated on a 6x6 maze using varying episodes and learning rates ($\alpha = 0.1, 1; \gamma = 0.9; 500 - 2000 episodes$). Dynamic Programming utilized value iteration (V^*) over 4000 iterations.

Start	1	0	1	0	0
Down	1	Right	Right	Down	1
Down	1	Up	1	Right	Down
Down	1	Up	0	1	Down
Down	1	Up	1	1	Down
Right	Right	Up	0	1	Finish

Figure 5. Optimal Path Learned by SARSA (Green Path).

We can see from figures 6 and 7 that if we increase the number of iterations, the lines indeed start converging, leading to the experiments becoming more valid. Plus when we increase the learning rates, the average Q-values at the terminal state nearly stay the same at a fixed number of iterations except for at the starting state where the average Q-value for learning rate = 1 has gone much lower compared to the other learning rate. With that kind of change, we can eventually deduce that the agent will learn more efficiently given the average q-values increasing dramatically through the maze path as the learning rate increases.

In figure 8, most of the V^* values converge when the number of iterations reaches at least 3000. However some of the V* values in some state like row (0,0) which the starting states need more iterations in order to converge. Therefore, our

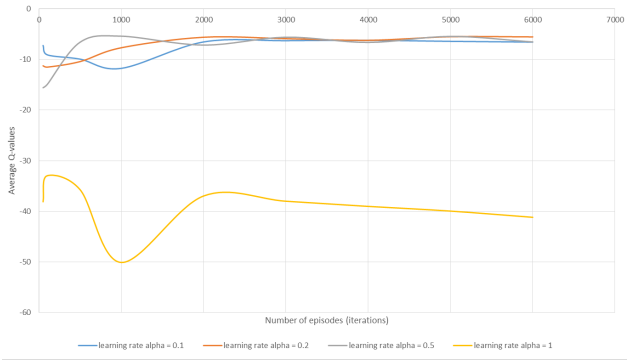

Figure 6. Convergence at Start State for SARSA (Various Learning Rates)..

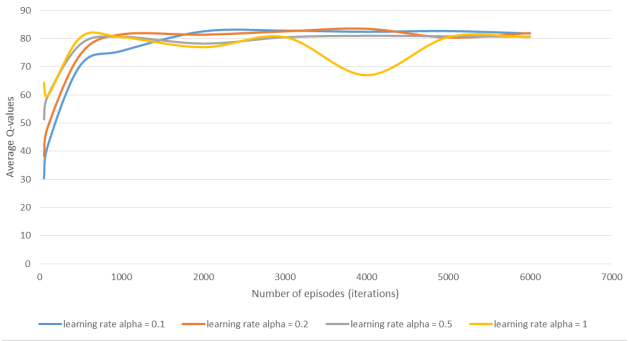

Figure 7. Convergence at goal State for SARSA (Various Learning Rates)..

best V^* results comes after performing 4000 iterationsIn figure 11, most of the V^* values converge when the number of iterations reaches at least 3000. However some of the V^* values in some state like row (0,0) which the starting states need more iterations in order to converge. Therefore, our best V^* results comes after performing 4000 iterations

In figure 9, each V^* values are positve and it traces the optimal path starting from the top left corner of the grid and going to every state with the maximum V^* value. Eventually, the agent will reach the termination state.

4. Literature Survey

4.1. Curiosity-Driven Exploration

The concept of curiosity-driven exploration has been explored in works by Pathak et al., where intrinsic motivation is used to encourage exploration in environments with sparse rewards. This aligns with our approach of introducing a curiosity factor to enhance exploration efficiency in Q-learning and Monte Carlo methods. Such intrinsic rewards help agents explore novel states more effectively, improving overall performance.

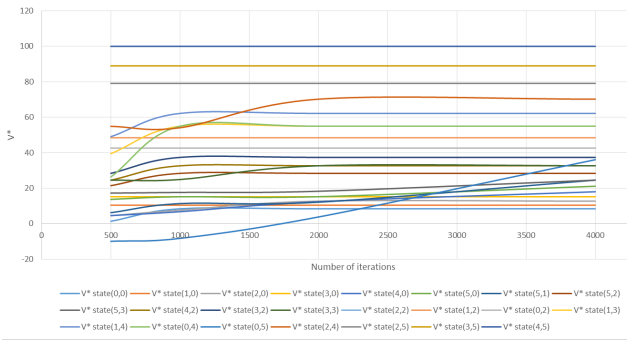

Figure 8. Graph showing the change in V^* values in every valid state as the iterations increase.(Dynamic Programming)

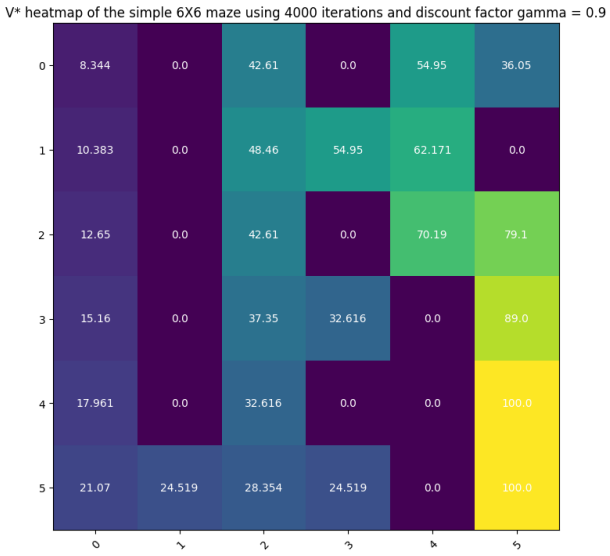

Figure 9. V^* Values Heatmap after 4000 iterations (Dynamic Programming).

4.2. Episodic Multi-agent Reinforcement Learning with Curiosity-driven Exploration

This article introduces a curiosity-driven exploration strategy in a multi-agent RL setting, enhancing coordination among agents and improving learning outcomes in complex environments.

4.3. Finite-Sample Analysis for SARSA with Linear Function Approximation

In (Shaofeng Zou, Temgyu Xu, Yingbin Liang, 2019) the researchers explains about how SARSA fits with reinforcement learning. It explains that "SARSA is an on-policy algorithm to learn a Markov decision process policy in reinforcement learning." This phrase does fit with the SARSA experiments performed in this projects. It inherits the formulas based on the Markov decision process involving the states, actions, probabilities, rewards, and discounts. Furthermore, the literature also explains more information on the linear function approximation.

4.4. Finite Time Bounds for Sampling Based Fitted Value Iteration

In (Szepesvari, Munos, 2005), , the article explains different forms of value iteration and how it originates from dynamic programming. It also discusses how Approximate Value Iteration (AVI) and Fitted Value Iteration (FVI) are guaranteed to converge. Evidence of AVI convergence in the article states that:

> "if $\gamma \in (0, 1)$ is the discount factor of the MDP and if A represents the operator that maps value functions to the space of functions of interest and if A is γ_0-Lipschitz with respect to the supremum norm, then the composite operator AT is a contraction provided that $\gamma\gamma_0 < 1$, since in discounted problems T is a contraction with contraction factor γ."

This statement aligns with the context of value iteration, where V^* is expected to converge to a point that leads to the optimal value.

5. Novelty

Our project introduces several novel elements to enhance the efficiency of maze solving using reinforcement learning:

- Curiosity Factor: We incorporate a curiosity factor into the Q-learning and Monte Carlo algorithms. This factor, denoted as λ, controls the weight of an intrinsic reward that encourages exploration. The intrinsic reward is calculated as $\lambda/\sqrt{N(s)+1}$, where $N(s)$ is the number of times state s has been visited. This reward is added to the environment's reward signal to guide the agent towards less explored states, improving exploration efficiency.
- SARSA: As the agent progresses through the maze and nears the termination state, the Q-values start to increase. The Q-values of all four actions near the terminal state are mostly greater than 50. We can deduce that if the Q-values along the optimal path increase, then the average Q-values in each optimal state will increase and by following the max average Q-values, it will lead the agent to the optimal path.
- Comparison of RL Algorithms: Our project compares the performance of Q-learning, Monte Carlo, SARSA, and Dynamic Programming on different grid sizes. This comparison provides insights into the strengths

and weaknesses of each algorithm in solving maze problems, contributing to the understanding of RL in complex environments.

6. Conclusion

Our experiments validate the effectiveness of RL algorithms in solving mazes efficiently: Curiosity-enhanced Q-learning/Monte Carlo improves exploration efficiency, SARSA benefits from increased learning rates/episodes, Dynamic Programming provides robust convergence using value iteration.

7. Future Work

Future work includes testing RL algorithms on larger mazes, incorporating deep RL techniques, and comparing performance against traditional search methods like BFS/DFS. We also wanted to experiment on altering the discount factor gamma. What factors of gamma gives us the higher reward and the optimal path at an efficient rate? Will the Q-values and the V* values converge faster to the optimal path?

8. Contributions

Sai Prasad implemented Monte Carlo and Q-Learning algorithms, conducting experiments to test hypotheses 1 and 3, while also reviewing literature section 4.1 and 4.2. Sandeep worked on Dynamic Programming and SARSA algorithms, performing experiments for hypotheses 2 and reviewing literature section 4.3 and 4.4.

References

Hannah, L. A. and Dunson, D. B. Approximate dynamic programming for storage problems. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011. URL https://icml.cc/2011/papers/235_icmlpaper.pdf.

Kim, J. and Cho, K. Attention-based curiosity-driven exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.10840*, 2019. URL https://arxiv.org/abs/1910.10840.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. URL https://arxiv.org/pdf/1705.05363.

Szepesvári, C. and Munos, R. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2005.

Zheng, L., Yang, J., and Zhou, Z.-H. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *arXiv preprint arXiv:2111.11032*, 2021. URL https://arxiv.org/abs/2111.11032.

Zou, S., Xu, T., and Liang, Y. Finite-sample analysis for sarsa with linear function approximation. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019. URL https://arxiv.org/pdf/1902.02234.

(Pathak et al., 2017) (Zheng et al., 2021) (Kim & Cho, 2019) (Zou et al., 2019) (Hannah & Dunson, 2011) (Szepesvári & Munos, 2005)