Markov Decision Processes

Brittany N. Tan
Georgia Institute of Technology, brittanytan@gatech.edu

***Abstract* – This project explores two different Markov Decision Processes (MDP), one small maze and one large maze. The smaller one is a simple gridworld problem with a simple path, while the larger one has 16 times the amount of states and is symmetrical in design to show exploration versus exploitation. The two gridworld problems are created using RL_sim. Each MDP problem is run and analyzed using value iteration and policy iteration, and Q-Learning.**

GRIDWORLDS

Two gridworlds will be analyzed throughout this project, one simple and one complex. The simple gridworld, as shown in Figure 1, contains 25 states with one goal state as denoted by the orange space. The bolded black lines denote walls. The agent has four possible actions at any state: up, down, left, and right, and its goal is to reach the goal state as quickly as possible. If the agent does not move, it receives a reward of -1. If it collides into a wall, it receives a reward of -50. There also exists a probability for each state that the agent does not go in its given direction. The same rules apply for the complex gridworld. This maze, as shown in Figure 2, is a 20 × 20 grid with a total of 400 states and five goal states.

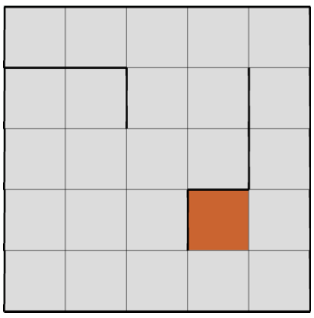

FIGURE 1
SIMPLE 5 × 5 GRIDWORLD

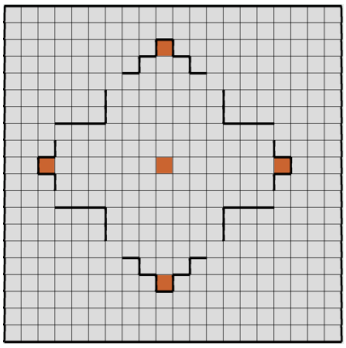

FIGURE 2
COMPLEX 20 × 20 GRIDWORLD

VALUE ITERATION

Value iteration is an algorithm that calculates rewards per state and makes decisions based on these rewards [1]. When the optimal policy is found, the algorithm terminates. For both mazes, the probability of unexpected action (PJOG) was altered and compared to the number of steps taken to converge.

As seen in Figure 3, both the small and large mazes follow similar patterns where they experience a steep increase follow by a steep decrease between probabilities of 0.6 and 0.9, and then converge at the end. The increase is expected—a greater chance of error will take the agent more iterations to reach its goal—but at 0.9 and 1, the agent converges with fewer steps. The reason for this is because of the reward calculated as it makes more unexpected decisions [2]; at a certain point, the cost of -50 for hitting a wall allows for the agent to converge towards its goal faster than to explore a path and then hit a wall. This explains why, regardless of complexity, both MDP problems converge towards similar values; however, the more complex problem always requires more steps due to a higher number of states.

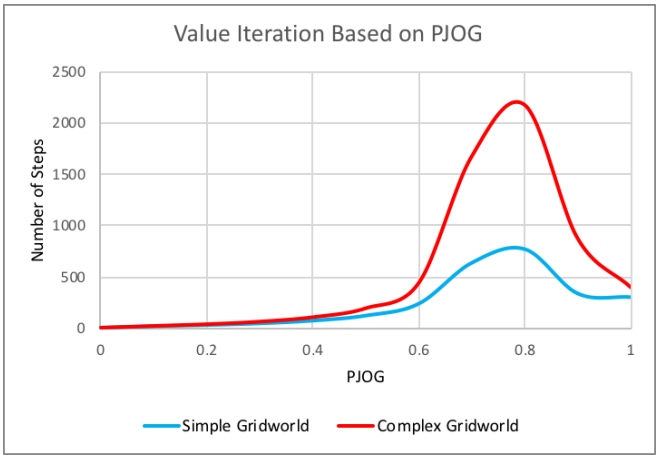

FIGURE 3
VALUE ITERATION VS. PROBABILITY OF ERROR PER STATE

Figure 4 displays the time taken for each maze as the probability changes. Both curves take similar patterns as the value iteration curves. Upon further analysis, when comparing the runtimes for each maze as shown in Figure 5, there exists a very strong linear relationship with an R^2 value very close to 1, showing that the runtime of value iteration is on the order of $O(n)$. Furthermore, analyzing the steps of the algorithm, each iteration calculates an immediate reward for each state and then moves to the next iteration which then calculates the reward for that state.

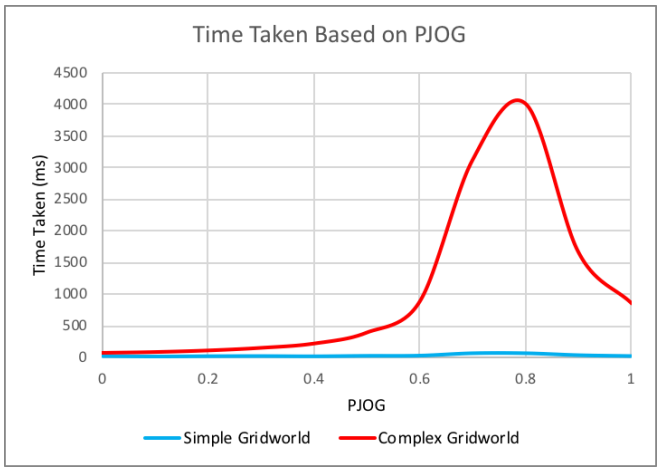

FIGURE 4
RUNTIME VS. PROBABILITY OF ERROR PER STATE

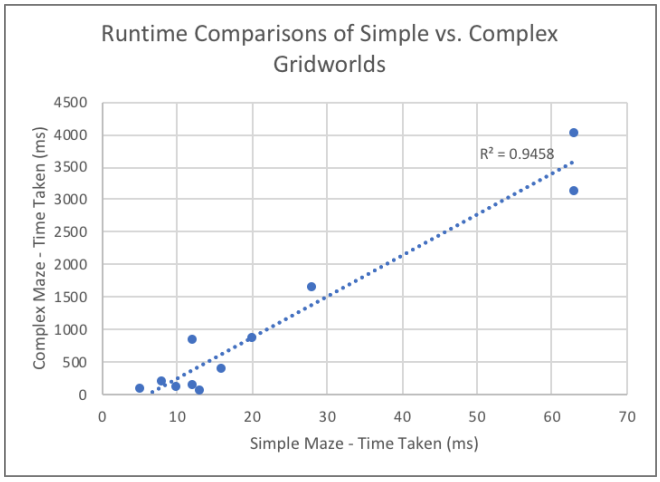

FIGURE 5
VALUE ITERATION SIMPLE MAZE RUNTIME VS. COMPLEX MAZE RUNTIME

POLICY ITERATION

This algorithm first evaluates the policy and then improves it [3]. This offers for a more holistic view of the policies rather than a greedy approach in value iteration. A visualization of each performance based on the probability of unexpected action is shown in Figure 6. Both the simple and the complex mazes experience decreases up to a certain PJOG, but then experience a spike. For the small maze, this happens at a PJOG of 0.9, while for the large maze, it happens at 0.8. An explanation for this divergence is the way the algorithm calculates rewards for each state. The algorithm operates as such [1]:

```
choose an arbitrary policy π'
loop
   π:=π'
   compute value function of policy π:
      solve for optimal value at iteration
   improve at each state, π'(s)
until π=π'
```

Unlike value iteration, policy iteration goes between a value determination step and a policy improvement step [3], so it calculates the long-term reward of all states in the MDP given an arbitrary policy. Because of this, with increase in PJOG comes significant increase in the amount of actions needed to evaluate, and while there exists a limited number of actions, these actions can be repeated infinitely, which is not the case in value iteration, explaining why that algorithm converges.

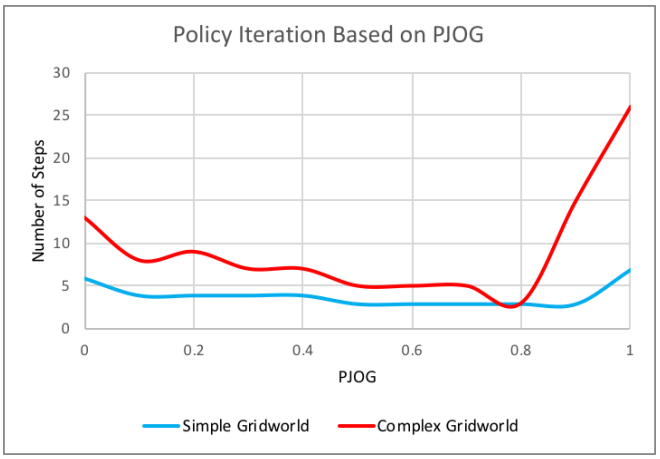

FIGURE 6
POLICY ITERATION VS. PROBABILITY OF ERROR PER STATE

Figure 7 displays a comparison of runtimes for each maze. Unlike the value iteration runtime plot which yields a high linear relationship, this plot yields a high polynomial relationship of degree 3. This indicates a polynomial runtime, specifically on the order of $O(n^3)$.

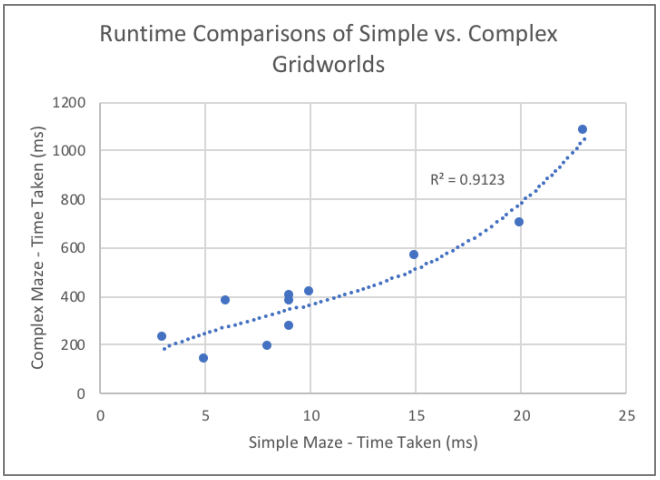

FIGURE 7
POLICY ITERATION SIMPLE MAZE RUNTIME VS. COMPLEX MAZE RUNTIME

Q-LEARNING

Q-learning is an iterative reinforcement learning algorithm where a Q value $Q(s,a)$ [1] is updated per iteration as in (1)

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max\big(Q(s',a') - Q(s,a)\big), \quad (1)$$

where α is the learning rate, r is the reward, γ is the discount factor, and $r + \gamma \max\big(Q(s', a') - Q(s, a)\big)$ is the overall learned value per iteration. Q-learning is a model free algorithm, meaning it does not assume a model of its environment [4]. Because of this, it is likely to perform poorer than value iteration or policy iteration, which can also be seen in Figure 8. Additionally, it requires many more iterations.

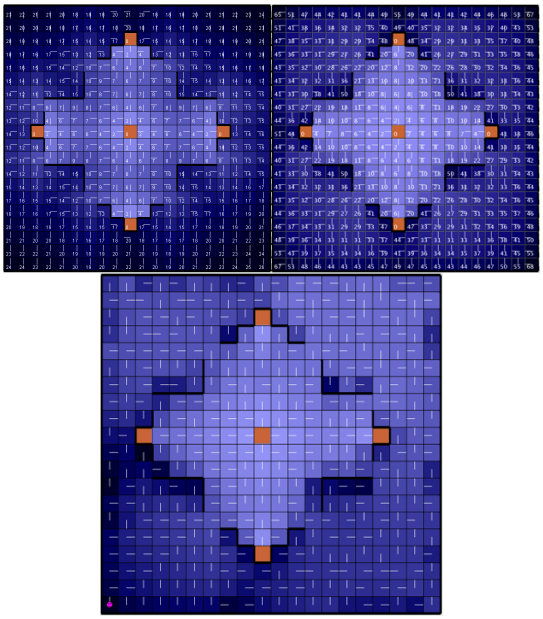

FIGURE 8
RESULTS: VALUE ITERATION (TOP LEFT), POLICY ITERATION (TOP RIGHT), AND Q-LEARNING (BOTTOM)

Figure 9 displays the simple gridworld before and after 100 Q-learning iterations. The pink smiley face represents the agent, and instead of taking the most direct path to the goal—right, right, right, up—it prefers taking a longer path with penalties. Regardless, Q-learning is said to be "exploration insensitive" [1], meaning that regardless of how the agent behaves, the Q-values will converge to optimal values.

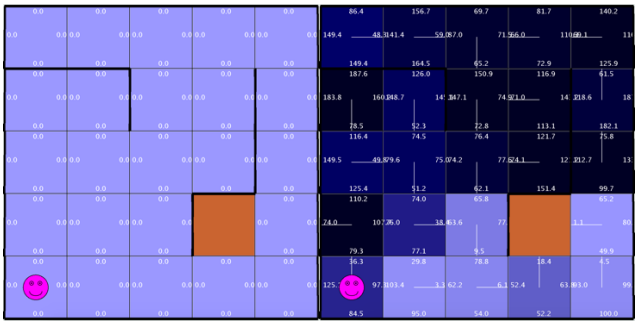

FIGURE 9
SIMPLE GRIDWORLD PROBLEM BEFORE (LEFT) AND AFTER (RIGHT) PERFORMING 100 ITERATIONS OF Q-LEARNING

CONCLUSION

From the results, value iteration and policy iteration both perform well, but require full domain knowledge of the state space. For deterministic models, this is not a problem. Of these two algorithms, value iteration is much more time efficient even though it requires many more iterations before converging, while policy iteration requires more time due to the policy improvement process. Nonetheless, value iteration would be most ideal for an MDP with a large state space. Q-learning takes far longer than either of the other two due to its need to both explore and exploit; however, it is model-free and requires no domain knowledge. Because of this, Q-learning is better adept to be used in real-life models.

REFERENCES

[1] Kaelbling, L.P., M.L., and Moore, A.W.(1996). Reinforcement Learning: A survey. *Journal of Artificial Intelligence Research*, 4:237-285

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. London, England: The MIT Press, 1998.

[3] Littman, M.L., T.D., and Kaebling, L.P.(1995). On the Complexity of Solving Markov Decision Problems. *UAI'95 Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 394-402

[4] A. Al-Tamimi, F. Lewis and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control", *Automatica*, vol. 43, no. 3, pp. 473-481, 2007.

AUTHOR INFORMATION

Brittany N. Tan, Undergraduate Student, College of Computing, Georgia Institute of Technology.