RL HW1 Report

Yu-Xiang, Luo

April 1, 2024

Q1

What methods have you tried for async DP? Compare their performance.

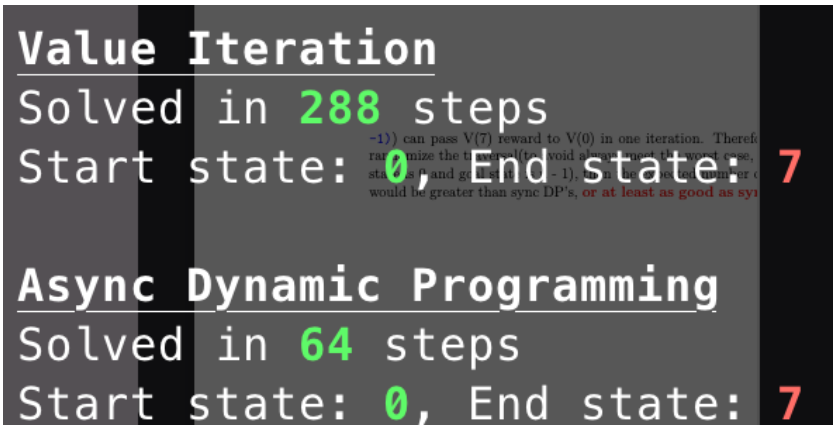
Unfortunately, I did not come out a way to implement prioritized sweeping and real-time DP. I can only perform the in-place DP.

- in-place DP: **Iteration steps: 968**, the value iteration sync DP is **1056** steps
- Prioritized sweeping: No implementation
- Real-time DP: No implementation

The reason why in-place DP is effective: Let's consider a simple scenario: a $n \times 1$ grid world, with S_{n-1} be goal state.

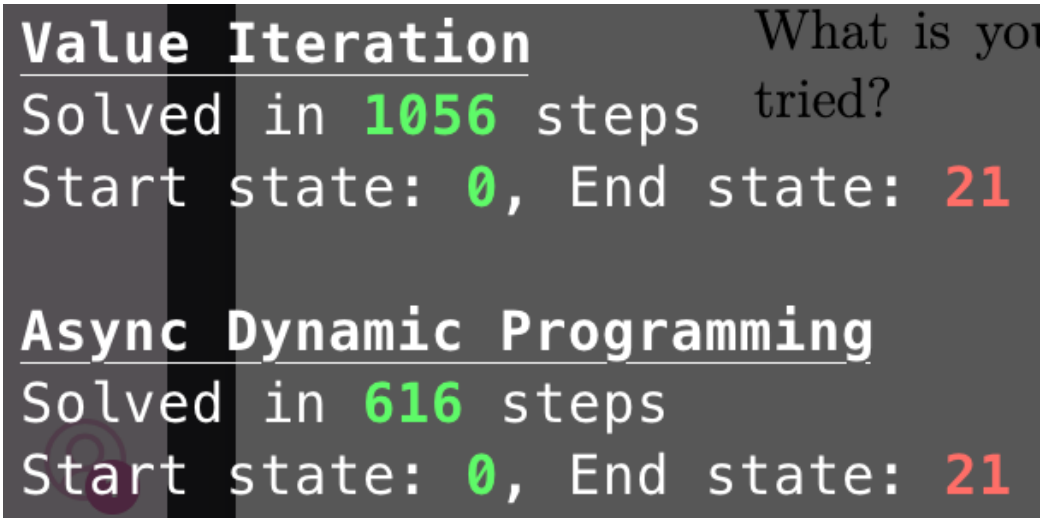
0	1	2	3	4	5	6	7

`step_reward = -1, goal_reward = 10`



The async DP use `[i for i in range(7, -1, -1)]` as Iteration order

This is a simple case, but I think is enough to explain my thought. Initailly, $\forall v \in V, v = 0$, which means in the first iteration, only V(7) is changed since `step(7, action).reward = 1`. While, at the second iteration, sync DP will only change the value of V(6) since only V(7) change. Then, V(5) and V(6), during next iterations. In other words, The reward of V(7) needs at least 7 iteratios to pass through V(0), yet in in-place DP, the best traverse order(`for i in range(7, -1, -1)`) can pass V(7) reward to V(0) in one iteration. Therefore, if we randomize the traversal(to avoid always meet the worst case, like start state is 0 and goal state is n - 1), then the expected number of update would be greater than sync DP's, **or at least as good as sync DP's**.(Sometimes it doesn't improve because the reward isn't good enough for the agent. Like the default reward for our HW1, step_reward = -1, goal_reward = 1 is not enough)



Randomize the order of traversal in every iteration, `goal_reward = 100`

I do try this method on the maze TA offer, and the result is as above.

Q2

What is your final method? How is it better than other methods you've tried?

Since we're allowed to store all step we've travelled. That is, we can store all `grid_world.step(state, action)`. Then, the worst iteration steps would be 4 * *state_space*.

By little improvement, step on goal state and trap state can be reduced to 1.