**National Taiwan University**
**Department of Electrical Engineering**
Name: An-Che, Liang
ID: B11705009

112-1 (Fall 2023)
Reinforcement Learning

# Assignment 1 Report

**Q1. What methods have you tried for async DP? Compare their performance.**

I have tried the three async DP algorithm mentioned by the professor.

## In-Place DP

This approach is to simply a modified version of value iteration algorithm. The difference is that prioritized sweeping update the state value right each state value estimations, rather than update all at once when the algorithm has visited all the states. I pass the example test-case with `step_count=968`.

## Prioritized Sweeping

This approach is guide the state selection with the biggest Bellman Error value through maintaining a prioritized queue (I use the built-in `heapq` module). The Bellman Error could be formalized as the following, function $Q'$ means a new state-action estimation.

$$|\max_{a \in A}(Q'(s,a)) - v(s)|$$

Once the algorithm has updated the current state, it looks for any other state that is affected by the current state, i.e. those state $s^*$ such that $\pi(s^*) = a$ and $s^* \xrightarrow[\text{a}]{} s$ with information of reverse state-action-next_state relationship. Then put the affected states into the heap with their own Bellman Error values.

However, after experimenting with several settings of with states should be inside the heap when the algorithm starts running, I find the approach of: run the algorithm until the heap is empty won't necessarily make our MDP converge, and since our goal is to find the optimal solution for all the possible states, this approach is thus not ideal. Instead for each iteration, I initialize the heap with every possible state and record the maximum $\Delta$ value just like value iteration, and terminate the algorithm if $\Delta$ reaches the threshold, but this approach makes the algorithm performs poorly in exchange for stability, and in the end I pass the example test-case with `step_count=1545`.

## Real-Time DP

This approach is to simulate a actual agent running in the environment and record the subsequent state values of each states, and choose the best action with estimated $Q'(s,a)$ value. One key aspect of this algorithm is to choose which state to put the agent in the beginning, and how many simulation to execute so the MDP can stably converge.

After several experiment with different random seeds and setup, I found out that RTDP could potentially have very high performance (converge at `step_count=600`), but wont converge under different random seed. The final solution is to randomly chose quarter of the possible state as start state for the virtual agents, and record the $\Delta$ value until observed convergence.I pass the example test-case with `step_count=1595`.

**Q2. What is your final method? How is it better than other methods you've tried?**

My final method is the In-Place DP method, though it's very simple, but it doesn't suffer from the stability problem like the other two asynchronous DP approach, which has to be solved by sacrificing performance, in the end those sacrifice out-grown their innovative algorithmic ideas.

Also, I have tried some methods to make sure MDP converges, such as record the states with the same $Q(s, a)$ value when executing different actions (signs of not converged MDP from my observation), but those methods does not work quite as well as the $\Delta$ value approach, thus are abandoned.