**National Taiwan University**
**Department of Electrical Engineering**
Name: Tu-Chin Chiang
ID: R11921038

112-1 (Fall 2023)
Reinforcement Learning

# Assignment 1 Report

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

I have experimented with in-place dynamic programming, prioritized sweeping, and real-time dynamic programming for asynchronous dynamic programming. However, in the case of real-time dynamic programming, I could not identify the appropriate method to access the starting state of the agent before the assignment deadline. Consequently, I had to resort to setting the state 0 as the start state and using the Bellman error to determine the end of the iteration. This led to variations in the step count depending on the initial state when employing real-time dynamic programming [1]. Additionally, in the implementation of prioritized sweeping, I opted to construct a priority queue rather than employing the algorithm outlined in the textbook [2]. Consequently, for smaller-sized mazes, the performance of prioritized sweeping was beaten by in-place dynamic programming and real-time dynamic programming.

The following are the step counts for each of the methods. Besides, all of the methods can find the optimal policy in the experiment.

| Method | Step count |
|---|---|
| In-place dynamic programming | 1056 |
| Prioritized sweeping | 1940 |
| Real-time dynamic programming | 1200 |

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

My ultimate choice for the method is a improved version of in-place dynamic programming. Specifically, the order of the updating determined by the Bellman error values of each state. As mentioned earlier, the real-time dynamic programming implementation I developed exhibited inconsistent optimal policy results. Consequently, in pursuit of a stable and effective approach, I opted for in-place dynamic programming as my asynchronous dynamic programming method.

Regarding the novel approach, I made noteworthy adjustments in two primary aspects: the policy improvement process and the number of states being updated. Firstly, instead of enhancing the policy after the value function converged, I updated the policy for each state during policy evaluation. It is important to note that this approach may not consistently yield the optimal policy, especially when employing a small discount factor. Secondly, I reduced the number of states updated in each iteration by excluding states with Bellman errors below a certain threshold. This reduction in interactions with the environment may lead to improved step counts. However, even though the performance of the novel approach was enhanced, to ensure the generation of the optimal policy, I ultimately chose a improved version of in-place dynamic programming as the preferred method.

# References

[1] Barto, A. G., Bradtke, S. J., and Singh, S. P. Learning to act using real-time dynamic programming. *Artificial intelligence 72*, 1-2 (1995), 81–138.

[2] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.