CS 747: Assignment 2

Mithilesh Vaidya
17D070011

1 Design and Observations

- The reward matrix and the transition probability matrix is initialised with 0s. Hence, any [state-action-next state] which is not specified is assumed to have 0 probability and 0 reward.
- In episodic MDPs, the transition probabilities and rewards for each end/terminal state-action pair is set to 0. This ensures that there is no singularity while solving for the value function using Bellman's equations.
- For linear programming, I used the CBC solver in Pulp and referred to [1] for formulating the MDP Planning as a linear programming problem.
- Termination condition for Value Iteration is: if the L1 norm between the value function and the new value function after applying the Bellman Optimality Operator is less than a threshold, the loop gets terminated i.e. exit the loop if:
$$||B^*[V^t](s) - V^t(s)||_1 < \delta$$
The threshold is currently set to $\delta = 10^{-10}$.
- For LP, the final value function is slightly different (differences in the last couple of decimal points) from the value functions computed using Value Iteration or Howard Policy Iteration (due to low precision of the LP solver). Hence, we calculate the action-value function from the value function, calculate the policy and then calculate the final value function using the policy.
- For HPI, the condition for Improvable Action: $Q^\pi(s,a) > Q^\pi(s,\pi(s))$ is sometimes erroneously evaluated to True (for some actions a) due to floating point precision errors. Hence, I modified the condition to:
$Q^\pi(s,a) > Q^\pi(s,\pi(s)) + \epsilon$ where the tolerance parameter $\epsilon = 10^{-10}$.

2 MDP formulation for Maze

- For the maze, each tile which is **not a wall** is considered a state. Adding states for walls will slow down the MDP Planning since time taken for Planning grows with the number of states.
- Rewards are as follows:
  - For bumping into a wall/taking an action which leads outside the maze: 0. In such cases, the agent remains in the same state.
  - For taking an action which moves the agent from one empty tile to another: 0. The agent changes state accordingly.
  - For reaching the terminal state: 10^5. Once it reaches the terminal state, any action keeps the agent in the terminal state and the reward for such actions is set to 0. (If it is non-zero, the value function for the terminal state will diverge)
- Discount factor γ is set to 0.9.
- Let t be the number of time steps it takes to reach the end state from the start state. Then, the total cumulative reward obtained by the agent is the reward at the last time step $= \gamma^t * 10^5$. Since this is a decreasing function of t, the agent will try to find a path which minimizes t, which is precisely our goal: finding the shortest path.
- Technically, we could have chosen any $\gamma \epsilon (0, 1)$. But if the number of steps required is large (in case of a very large maze), γ^t will decay quickly and we may run into finite precision problems. On the other hand, a γ very close to 1 may slow down the algorithm since the contraction factor is very high. Also, the reward could have been any positive number but setting it to a high value such as 10^5 may help in preventing floating point underflow errors.

References

[1] Linear Programming With Python and Pulp, `https://benalexkeen.com/linear-programming-with-python-and-pulp/`