CS 747
Assignment 2-Report

17D170011, Siddhesh Pawar

1. Task 1

For continuing tasks, the list of end states is kept empty(when the input is -1)

For Value Iteration:

A list of nonterminal states is maintained for each task which is the normal list of states for continuing tasks and excludes the terminal(end states) in case of episodic tasks. The purpose of maintaining such a list is while carrying out iterations, the transitions from terminal states can altogether be avoided thus making the algorithm more computationally efficient

The tolerance, that is the value of epsilon has been set to 10^(-20) with the number of iterations being equal to 50000. The transitions are stored in a dictionary, that makes the performance considerably faster, the dictionary has not been used in LP and Howard because of ease of using arrays that represent the constraints.

For Howard policy iteration:

The tolerance, that is the value of epsilon has been set to 10^(-10). The function bellman_equations has been defined to handle the matrix operations that occur during the solving of bellman equations.

For episodic tasks, the value of terminal states is set to zero during the policy evaluation and thereafter the policy improvement step is carried out until the stable policy is found that is the old and new values(after the iteration) have a difference less than epsilon

For Linear Programming:

The default solver that is PULP_CBC_CMD is used to solve the LP Problem and has been explicitly mentioned so that it surpasses the output

For episodic tasks, extra constraints have been added to the LP problem.

Once The LP is solved, we then solve the Bellman's equations.

The execution time for all the above three is less than a minute(8 seconds max for each of the given test cases)

2. Task 2

Assumptions:

The discount factor is taken to be 0.9999 for the maze as the MDPs have an end states and the task can be viewed as episodic

For the encoder, the walls haven't been assigned any state, in order to reduce the number of states, and then transitions have been defined accordingly. The maze is taken as an input and state allotment is done according to the numbers of the maze(that is segregating the positions in the gird as end states, start states, and valid moves)

The valid actions are encoded in the form of coordinates of the grid which are stored as a dictionary

The reward of reaching the designated end state is kept to be 1000. This perfectly depicts an example of MDP where the current actions have a reward at the end. The high value of gamma also suggests that the MDP is far-sighted

The action taken at each state which is the output of the planner is encoded as a direction based on the change in coordinate and the set of actions is the output by the decoder.

The fastest results are achieved by using Value iteration while the slowest are achieved by running HPi, linear programming performs fast for the smaller grids.