Reinforcement Learning CIA-2 assignment

Problem:

Create a 100x100 grid with obstacles in between 2 random points. Build an MDP based RL agent to optimise both policies and actions at every state. Benchmark DP method with other RL solutions for the same problem.

Solution:

Given a grid of size 100 x 100, let us assume we are starting at (1, 1) and our goal is to reach (100, 100). We will try to represent the grid as a 2-D array with dimension, (100,100). Totally the grid contains 10000 cells. An obstacle and space are marked as 1 and 0 respectively in the grid.

To create the obstacles in the grid,

1. Random Obstacles: Randomly place a specific number of obstacles across the grid.
2. Path Constraints: Ensure there is always a clear path from the start (1, 1) to the goal (100, 100).
3. Obstacle Placement with Path Constraint (place\_obstacles\_with\_path):
   1. For each random obstacle placement, it temporarily places an obstacle and checks if the path is still available.
   2. If placing the obstacle blocks the path, it removes it and tries a new position.
   3. This process continues until we successfully place the specified number of obstacles while maintaining path connectivity.
4. Path-Checking Function (is\_path\_available):
   1. This function uses BFS to check if there’s still a path from (0, 0) to (99, 99).
   2. It starts from the top-left corner and attempts to reach the bottom-right corner, marking visited cells to avoid revisiting them.
   3. If it reaches the goal, it returns True; otherwise, it returns False

Possible states and actions:

Total\_possible actions(8): (x-1,y-1),(x,y-1),(x+1,y-1),(x-1,y),(x+1,y),(x-1,y+1),(x,y+1),(x+1,y+1)

The position of the agent is represented by (x,y) co-ordinates and its respective state s.

Normal\_state: (x,y): X!=1 x!=100, y!=1 y!=100

Actions: (x-1,y-1),(x,y-1),(x+1,y-1),(x-1,y),(x+1,y),(x-1,y+1),(x,y+1),(x+1,y+1)

Start\_state = (1,1) :-

possible actions from this state: (x+1,y), (x+1,y+1), (x,y+1)

Goal\_state = (100,100) :-

No further actions, terminate the episode.

Obstacle state = (x,y)

Actions: Go to previous state position.

Top wall state = (1,y):- y!=100 and y!=1.

Actions: (x+1,y), (x+1,y+1),(x+1,y-1),(x,y+1),(x,y-1)

Right wall state = (x,100) x!=1,x!=100

Actions: (x-1,y),(x+1,y),(x-1,y-1),(x+1,y-1),(x,y-1)

Left wall state: (x,1), x!=1 and x!=100

Actions: (x-1,y),(x+1,y),(x-1,y+1),(x+1,y+1),(x,y+1)

Bottom wall state: (100,y) y!=1 and y!=100

Actions: (x-1,y), (x-1,y+1),(x-1,y-1),(x,y+1),(x,y-1)

Bottom-left corner: (100,1)

Actions: (x-1,y),(x-1,y+1),(x,y+1)

Top-right corner: (1,100)

Actions: (x+1,y),(x+1,y-1),(x,y-1)

Total types of states = 10, total\_possible actions = 8

State representation: a three number list -> [x,y,s]

Where x and y denotes the grid position of the agent and s denotes the type of state of that cell.

Example: cell (1,1) -> [1,1,start\_state]

We consider each position in the grid to be a state and then we try find the optimal path by assigning the best action at each position in the grid such that it takes you to the goal while maximizing rewards.

Hence our total no, of states = no. of cells in the grid = 10000

Reward Model:

For obstacle state: -20

For Goal state = +250

For normal states = +1/no\_of steps taken so far (To induce model to learn a path with minimal steps)

Optimization: Move from start\_state to goal\_state in minimal number of steps or actions and avoiding obstacles in the path. i.e, find the shortest path from start\_state to goal\_state.