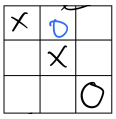
IE 5331-003 Homework 4

Spring 2025
Due Date: Sunday, Apr-13-2025, 11:59 PM

Problem 1

Remark: This is a typical finite-horizon deterministic Markov Decision Process (MDP) problem. In the first part, you will explore how actions and transitions in an MDP define an enumeration tree of all possible future trajectories. In the second part, you will use backward induction to propagate terminal values back through the tree, updating the value of each preceding state. Together, these steps allow you to identify the optimal policy and state values. Conceptually, this approach applies to all finite-horizon MDPs, whether deterministic or stochastic: (1) use actions and transitions to construct an enumeration tree, and (2) backtrack from the terminal nodes to compute optimal values and actions at earlier states, eventually reaching the root (initial state). The size of the tree often determines the problem's complexity. Therefore, merging identical or symmetric states is a common strategy for improving efficiency. In practice, merging similar (but not identical) states leads to approximation methods.

Problem: The popular two-player game of Tic-Tac-Toe involves players alternately placing their symbols on a 3x3 grid. The first player (P1), who uses crosses, and the second player (P2), employing circles, each aims to align three of their pieces in a continuous line—vertically, horizontally, or diagonally. Victory is claimed by the player who first achieves this alignment. The game results in a draw if the grid is filled without either player forming such a line. Let W, L, and D denote the possible outcomes of winning, losing, or drawing for the first player P1 (e.g., P1 prefers W, and P2 prefers L). Currently, P1 is facing the following game state (let's call this the *initial state*) after four steps of the game.



Using backward induction, answer the following questions.

(1). Starting from the initial state, identify all the possible actions for P1, and determine the next state associated with each action.

(2). For each new state, do the same steps for P2.

(3). Keep the process until every game ends up with a determined value (W, L, or D).

(4). Perform a complete backward induction on the above enumeration tree, and determine the best move for each player at each state. Which action is the best for $P1$ at the initial state?

Problem 2

Develop a shortest path algorithm using backward induction. Randomly generate five network instances and solve them using this developed algorithm. Report the length of the shortest path from each vertex to the terminal t.

Problem 3 [Grid World Game]

In the following grid world, a player aims to maximize their total discounted reward by choosing a direction to move at each cell.

0	0	0	1
0		0	-100
0	0	0	0

Each time the player enters a cell, they receive the reward shown in that cell. The player can attempt to move up (↑), down (↓), left (←), or right (→) from any cell. However, if the movement would take them outside the grid or into a wall (the black cell), they remain in the same position (i.e., they "bounce back").

Actions are stochastic: each intended move has an 80% chance of succeeding, but there is a 10% chance of drifting to the left and a 10% chance of drifting to the right relative to the intended direction. For example, if the player chooses to move right (→), there is an 80% chance of moving right, a 10% chance of drifting up (left relative to the intended direction), and a 10% chance of drifting down (right relative to the intended direction). Based on this setup, complete the following problem.

(1). Design the set of states, set of actions, reward function, and state transition function.

(2). Write down the associated Bellman equation.

(3). Find the optimal policy using the policy iteration method. Provide the associated python code.

(4). Try different discount factor γ, see how would this affect the optimal policy.

(5). Choose 10 iterations of your implementation to show the evolution of the associated value and policy functions.