MDL Assignment 2

Linear Programming

Part 3

- Dayitva Goel: 2019101005
- Prajneya Kumar: 2019114011

1 Overview of the Linear Programming

Linear Programming is an optimization algorithm to find the best outcome in a Markov Model where the objective function and constraints are represented by linear relationships.

We wrote a program that solves the Indiana Jones and MM game we had been provided. Our code outputs the following:

1. The A Matrix
2. The R array
3. The alpha array
4. The optimised x array
5. The derived deterministic policy
6. The finalized objective value

2 Building the A Matrix

To build the A Matrix, we populated it column wise. While doing this, we noted each state; and for each of these states, we calculated the valid actions that can be taken, thus arriving at (state, action) pairs. For each of these (state, action) pairs, we populated the corresponding cell of the A Matrix with respective probabilities.

Since all inflow probabilities need to be subtracted and all outflow probabilities need to be added, we initialised each cell corresponding to a (state, action) pair of the A matrix as 1, and subtracted the probabilities in the cell corresponding to (next_state, action). This also takes care of self loop, as probabilities of self loop are cancelled out in this method.

We obtained the A-Matrix of the shape (600, 1936) as we have a total of 5 * 3 * 4 * 2 * 5 (position, material, arrows, state of MM, health of MM respectively) = 600 states and 1936 valid pairs of (STATE, ACTION).

3 Finding the Policy

After obtaining the optimised x array via the pre-defined functions of the `cvxpy` library, we note that x_{ij} denotes the number of times action j is taken in state i. Using this number, we take note of the maximum value of x_{ij} over all j and choose the corresponding j as the best policy for state i; thus constructing the entire policy.

Analysis

When IJ is in North Square:

> If MM is in Dormant State, IJ moves down to Center Square to take an attacking position. If IJ has material, he chooses to craft arrows, else prefers to stay in North Square.

When IJ is in East Square:

> IJ choses to HIT or SHOOT depending upon the number of arrows he has.

When IJ is in South Square:

> IJ decides to stay or gather, if MM is in the ready state.
>
> IJ decides to move up to Center Square to take an attacking position if MM is in dormant state.

When IJ is in West Square:

> If IJ has arrows, he shoots. If he doesn't, he usually decides to STAY out of MM's attack or go RIGHT to gain a better attacking position.

When IJ is in Center Square:

> If MM is dormant, IJ takes LEFT or UP action to escape. He usually chooses to craft more arrows by moving up if he has materials.
>
> If MM's health is 25 and is in dormant state, IJ chooses to shoot.

4 Alternative Policies

We must also note that multiple policies can be obtained for the same Markov Decision Process (MDP). We claim that an alternative policy can also exist because:

1. As explained earlier, we chose the optimal policy by choosing the action which has been chosen as the action maximum number of times. It is possible that this value of *maximum choice* can be the same for two actions. In this case, we are branched with two alternatives policies to proceed with here. However, this would not affect the values of of A matrix, R array, alpha aray etc.
2. If we choose a different start state than the current one, we will end up with an alternative policy. However, this will change the value of **the alpha array**.
3. If we use a different set of rewards, punishments (penalties), or step costs, **the R array** will change and thus lead us to a different policy.