CS312: Aritficial Laboratory Lab Assignment 8

COMPILED BY MEHUL BOSE (170030010)

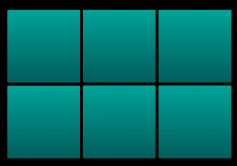
Description of the problem domain:

In the Raniganj Coalfields of Asansol, Coal miners are faced with a difficult challenge. The coal miners are tasked with the exploration of a coal field, which is 3 cells across and 2 cells deep. Each cell contains a certain amount of coal. The miners are faced with a choice of which of the top cells they can dig up. Upon digging up a given cell, the miners incurs a negative reward proportional to the depth of the cell, as digging deeper incurs more cost. Each cell, upon being dug, returns a positive reward proportional to the amount of coal present in the cell. The coal miners' actions are picking out which of the four cells to dig into. Upon the miners' choice, the digger has an P chance of digging correctly and 1-P chance of being unable to dig the given hole. The latter will result in a negative reward but no positive reward. Like every MDP, it is defined by a tuple of (S, A, P, T, N).

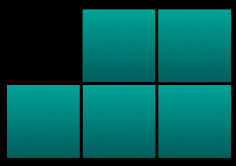
States of the MDP (S):

Each configuration of remaining cells is a state. For example

State 0 is:



State 1 is:



Actions of the MDP (A):

Action 0: Hold status. Do not dig
Action 1: Dig into leftmost cell. If bottom is reached then hold.
Action 2: Dig into centre cell. If bottom is reached then hold.
Action 3: Dig into rightmost cell. If bottom is reached then hold.

State Transition Function (P):

This function shows the next state given the current state and the next action.
In this MDP, that is given as the environment class' attribute nextstate.

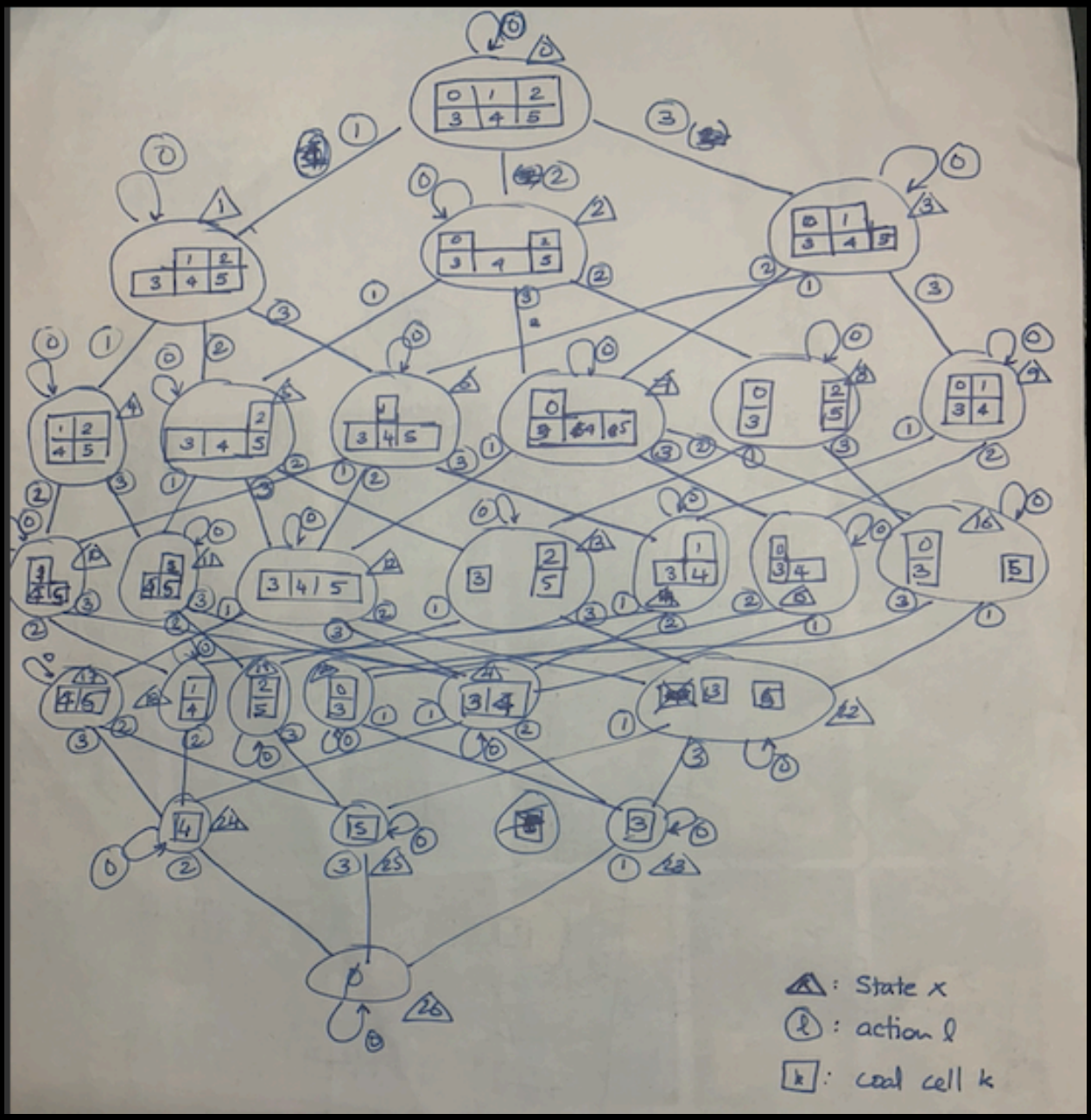
Reward Function (R):

This function shows the cost of making a transition, given a current state and the next action.
In this MDP, the reward function gives 0 if the current state and next state (given by executing next acton on the current state) are the same. Otherwise it gives the cost gained by digging up the coal subtracted by the cost of running the digger

Discount Factor (N):

This factor controls the importance of future rewards in comparison to immediate rewards.

State Transition Diagram



Input format:

Grid coal values (2 lines with 3 integers separated by a whitespace)
Probability of success (between 0-1)
Cost of digger running
Grid coal to cost conversion ratio

Example 1:

7 4 2
8 5 1
0.8
3
1

Optimal Policy Check

In this setting, it is obvious to see that the optimal policy would be to dig up the leftmost and centre pile to the bottom, while the rest can contain more 0s, 1s and 2s (in the policy) but no 3s at all as they lead to loss.
Policy obtained from value iteration:
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 2. 1. 1. 1. 1. 1. 2. 2. 0. 1. 1. 1. 1. 2. 0. 0.]
Policy obtained from policy iteration:
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 2. 1. 1. 1. 1. 1. 2. 2. 0. 1. 1. 1. 1. 2. 0. 0.]
It is clearly visible that the policy is similar to the optimal policy

Example 2:

7 4 4
8 1 6
0.8
3
1

Optimal Policy Check

In this setting, the optimal policy would be to dig up leftmost and rightmost cells to the bottom, but leaving the centre cell with only one dig
Policy obtained from value iteration:
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 3. 3. 1. 1. 1. 1. 1. 3. 2. 3. 1. 1. 1. 1. 0. 3. 0.]
Policy obtained from policy iteration:
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 3. 3. 1. 1. 1. 1. 1. 3. 2. 3. 1. 1. 1. 1. 0. 3. 0.]
It is clearly visible that the policy is similar to the optimal policy

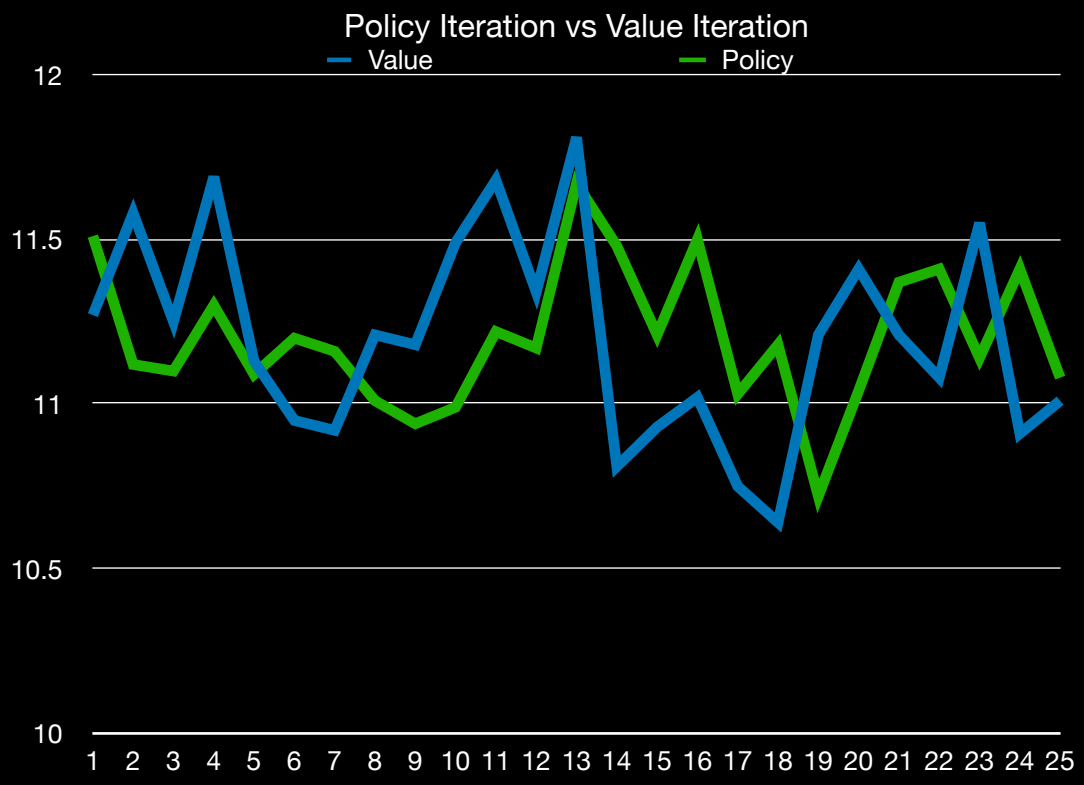
Example 3:

7 4 2
8 5 7
0.8
3
1

Variations in Discount Factor

Value iteration:
With N = 0
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 2. 1. 1. 1. 1. 1. 2. 2. 0. 1. 1. 1. 1. 2. 0. 0.]
With N = 1
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 3. 2. 1. 1. 1. 1. 1. 3. 2. 0. 1. 1. 1. 1. 2. 3. 0.]

Policy iteration:
With N = 0
[1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 2. 2. 1. 1. 1. 1. 1. 2. 2. 0. 1. 1. 1. 1. 2. 0. 0.]
With N = 1
[1. 1. 1. 3. 2. 1. 3. 3. 1. 1. 3. 2. 3. 1. 1. 1. 3. 3. 2. 0. 1. 1. 3. 1. 2. 3. 0.]



Policy Iteration vs. Value Iteration

Policy Iteration	Value Iteration
Policy Iteration iterates over the policy which is constructed	Value iteration iterates over the value which is constructed.
To iterate over policies, the algorithm sets an initial policy, finds it's value, then extracts a new policy from the obtained value.	To iterate over values, the algorithm first generates an initial set of values, and modifies them according to the reward system until concurrent readings are obtained.
Policy iteration is faster as a policy converges faster than a value function	Value iteration is slower in comparison
It gives slightly less optimised results over value iteration	It gives slightly better worse results than policy iteration

Conclusions:

Both Policy iteration and value iteration are effective methods of determining of a policy for a Markov Decision Problem. However, as is the case always, value iteration is more time consuming but gives more optimum results over the policy iteration. For this particular problem, I would recommend using a policy iteration as the difference in time is not particularly noticeable, and the difference created by the time consuming value iteration method is not really visible. However as the coal mine increases in size, policy iteration would be preferred due to the large increase in time for value iteration.

The difference is not particularly visible in this particular question