CS 747: FILA

Vishruth N
Roll: 200010088

PART 1 - Implementation of various methods to solve a given MDP

I have simply mentioned pseudocodes for my algorithms because task 1 was more like an implementation of the algorithms and further description didn't seem required.
Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto, 2nd edition, MIT Press, 2018, served as the source for task 1's solution. I have implemented the Linear Programming, Value and Policy iteration methods as shown below. The following book http://incompleteideas.net/book/RLbook2020.pdf, and course slides served as helpful resources while completing my task 1

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(terminal) \doteq 0$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

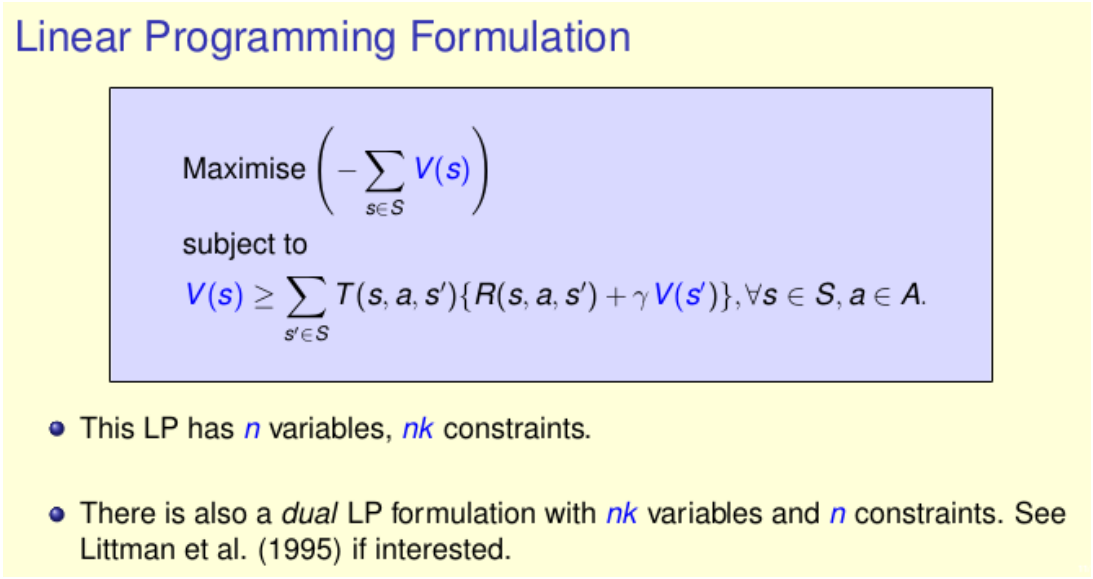
Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \text{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

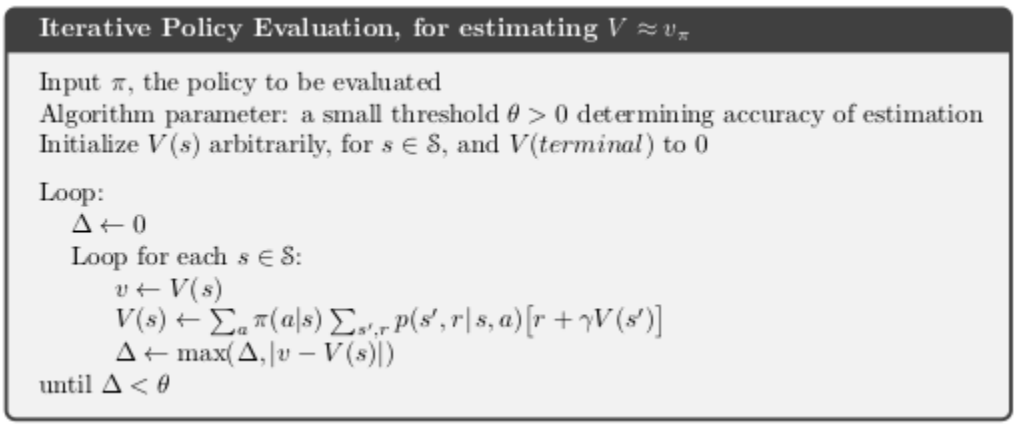
Linear Programming



I learned the use of PuLP and implemented it in my algorithms using the following resource: https://www.coin-or.org/PuLP/index.html, https://stackoverflow.com/questions/64005808/pulp-solvers-python-3-8-spyder-4

Next, for the policy iteration part, I have used the following algorithm represented in pseudocode, as shown.

Policy Evaluation



TASK 2 - Modeling the Cricket game as an MDP

In the encoder.py, I have initialized various helper functions to model the transitions and generate next states. I have also created an extended list of states which specify which batsman is on strike and whether the game is over. The list of endstates comprises of the states where the number of balls/runs remaining reaches 0, or when a batsman gets out.

The transit() function calculates the next state when the current state and outcome is specified according to the logical rules of the cricket game.