CS747: Programming Assignment PA-3

Deshpande Varad Shailesh, 21D070024

November 5, 2023

1 Initial Steps

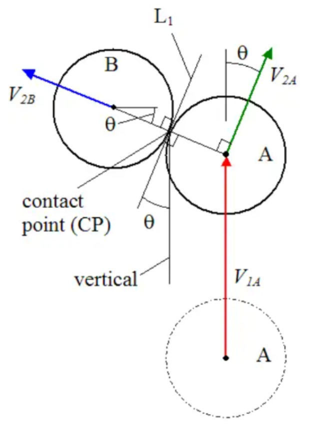

Figure 1: Optimal Angle

Initially, I tried calculating the correct angle that would put the ball into the hole, with some deterministic forces depending on the distance between the ball and hole, ball and the cue.
The initial angle is calculated using the coordinates of centre of final position of cue ball after finding theta and taking cases for all quadrants. This seemed to work well in the initial levels, but very quickly, some problems became apparent:

- Grazing Cue shot such that the cue is horizontal and the ball has to traverse vertically. Cue barely hits the ball and the action keeps oscillating.
- Too much force applied in unforeseen situations that causes the ball to bounce off the protrusions around the hole, messing up the build up toward the hole.

To choose the ball-hole pair:
Policy **P1: Choose the ball that is closest to any hole.**

2 A step forward: Look-up of the next state

Having tried hard to tune the force parameters, I realised that I could not get a better score without a look-up. I tried the following ideas:

- With **P1**, Fix the optimal angle through calculation, and search through a set of 5 forces = [0.2,0.3,0.4,0.5,0.65]
- With **P1**, Fix an angle and forces = [0.25,0.45,0.55,0.65,0.85] with optimal angle for all except 0.85 which has a head-on angle with the target ball. The head-on was meant to remove a deadlock situation, and bring the cue ball/target ball into a better position to score.

I was averaging each force's outcomes over three samples.

While both the above methods proved to be a slight improvement over the hand-tuning policy, they extracting very little information after performing such a heavy simulation and I thought that I could make more out of these simulation.

3 The Final Policy

Therefore, in final submission, I use the following policy:
forces = [0.15, 0.2,0.3, 0.45, 0.6, 0.7]

- Simulate for every force, for every available ball, the next state, with the optimal angle
- Choose that ball which goes closest to the closest hole from the ball in the present situation.
- If the shot is a grazing shot, choose the next optimal ball
- If the shot is still a grazing shot, then:
  - angle = angle_optimal*0.6 + numpy.random.randint(0,10)/50.0
  - force = 0.37

The look-up helps to get the most out of the present situation of the cue. The last step in the policy helps to remove the cue from a strategically weak position.

Taking a large-scale decision in this way helps me steal the luck in situations where the ball rebounds off the boundary viz., coming in a better position to be put.