Matt Dickenson mcd31
CS570 — Fall 2014 **Homework 5** Due: November 25, 2014

Problem 1

A graphical version of the MDP described in the homework is shown in Figure 1:

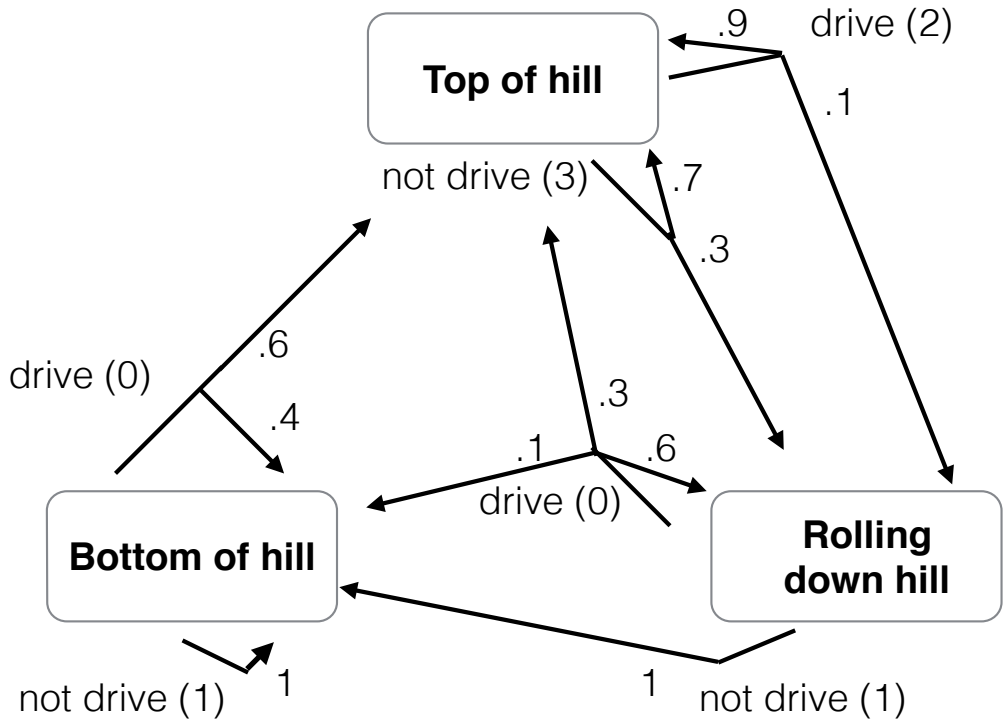

Figure 1: Graphical MDP

I also set up the transition matrices and utilities for each action for use in later problems:

Listing 1: R Code for Transition Probabilities and Utilities

```
# Set up transition matrices
drive      = matrix(NA, nrow=3, ncol=3)
not.drive = matrix(NA, nrow=3, ncol=3)
states   = c("top", "rolling", "bottom")
actions = c("drive", "not.drive")

rownames(drive) = rownames(not.drive) = states
colnames(drive) = colnames(not.drive) = states

drive["top", "top"]          = .9
drive["top", "rolling"]      = .1
drive["top", "bottom"]       = .0
```

```
13 drive["rolling", "top"]      = .3
14 drive["rolling", "rolling"] = .6
15 drive["rolling", "bottom"]  = .1
16 drive["bottom", "top"]       = .6
17 drive["bottom", "rolling"]  = 0
18 drive["bottom", "bottom"]   = .4
19
20 not.drive["top", "top"]           = .7
21 not.drive["top", "rolling"]       = .3
22 not.drive["top", "bottom"]        = .0
23 not.drive["rolling", "top"]       = 0
24 not.drive["rolling", "rolling"] = 0
25 not.drive["rolling", "bottom"]  = 1
26 not.drive["bottom", "top"]        = 0
27 not.drive["bottom", "rolling"]  = 0
28 not.drive["bottom", "bottom"]   = 1
29
30 # Set up utility matrix
31 utils = matrix(NA, nrow=3, ncol=2)
32 rownames(utils) = states
33 colnames(utils) = actions
34 utils["top", "drive"]           = 2
35 utils["top", "not.drive"]       = 3
36 utils["rolling", "drive"]       = 0
37 utils["rolling", "not.drive"] = 1
38 utils["bottom", "drive"]        = 0
39 utils["bottom", "not.drive"]   = 1
```

Problem 2

My $\mathcal{R}$ code for value iteration is shown below. The values were initialized to zero. The policy recommended does not change after the fourth iteration, but it took five iterations for the values to become stable.

Listing 2: R Code for Value Iteration

```
41 delta     = 0.8
42 max_iters = 25
43 v = matrix(0, nrow=length(states), ncol=max_iters)
44 rownames(v) = rownames(a) = states
45 options = list("drive" = 0, "not.drive" = 0)
46
47 iters = 2
48 while (iters <= max_iters) {
```

```
49   for (state in states){
50     for(action in actions){
51       options[action][[1]] = utils[state, action]
52       for(next.state in states){
53         cmd = paste(action, '["', state, '", "', next.state, '"]', sep="")
54         p = eval(parse(text=cmd))
55         options[action][[1]] = options[action][[1]] +
56           delta * p * v[next.state, iters-1]
57       }
58     }
59     o = unlist(options)
60     v[state, iters] = max(o)
61   }
62   iters = iters + 1
63 }
```

The recommended policy is as follows:

State	Best action
Top of hill	Not drive
Rolling down hill	Not drive
Bottom of hill	Drive

The long-run value of not driving when at the top of the hill is about 7.05 after the fifth iteration. The long-run value of driving when rolling down the hill is about 3.35 after the fifth iteration. The long-run value of driving when at the bottom of the hill is 3.91.

Problem 3

When solving for the optimal policy using policy iteration, we arrive at the same policy as in (2) after three iterations but the values are not the same until the fourth iteration (it took until the fifth iteration in Problem 2 because v_0 was initialized to zero rather than the values of the "not driving" policy). The best policy in the last two rounds (v_0 and v_1) is "never drive" but the optimal long-run policy is:

State	Best action
Top of hill	Not drive
Rolling down hill	Not drive
Bottom of hill	Drive

and again the values are 7.05 at the top of the hill, 3.35 when rolling down the hill, and 3.91 at the bottom of the hill.

The $\mathcal{R}$ code for policy iteration is very similar to value iteration, with the addition of a matrix a to keep track of the `argmax` at each iteration:

Listing 3: R Code for Policy Iteration

```
delta      = 0.8
max_iters = 25
v = matrix(0, nrow=length(states), ncol=max_iters)
a = matrix(NA, nrow=length(states), ncol=max_iters)
rownames(v) = rownames(a) = states
options = list("drive" = 0, "not.drive" = 0)

a[ , 1] = rep(which(actions=="not.drive"), 3)
v[ , 1] = utils[ , 2]

iters = 2
while (iters <= max_iters) {
  for (state in states){
    for(action in actions){
      options[action][[1]] = utils[state, action]
      for(next.state in states){
        cmd = paste(action, '["', state, '", "', next.state, '"]', sep="")
        p = eval(parse(text=cmd))
        options[action][[1]] = options[action][[1]] +
          delta * p * v[next.state, iters-1]
      }
    }
    o = unlist(options)
    v[state, iters] = max(o)
    a[state, iters] = which(o==max(o))
  }
  iters = iters + 1
}
```

Problem 4

Changing discount factor Decreasing δ to $\delta = 0.5$ changes the optimal policy to "never drive, regardless of the state." When the future is discounted at this rate (i.e. we care more about the present than in the example given in the homework), there is less incentive to expend energy trying to get back to the top of the hill. The long-run equilibrium of this strategy is to remain at the bottom of the hill, content to receive 1 unit of energy per time period.

Changing transition probabilities If we change the transition probabilities for "not drive" in the "rolling down the hill" state so that $p(s' = \text{rolling}|a = \text{not drive} \,\&\, s = \text{rolling}) = 1$, the optimal policy becomes "not drive if at the top of the hill, and drive if rolling or at the bottom of the hill." This makes sense intuitively, because if we get stuck rolling down the hill unless we drive, it makes sense to drive and have a chance of getting back to the top of the hill (where we glean the most energy).

Changing reward for an action I propose changing the reward for not driving when at the bottom of the hill from 1 to 3. The optimal policy would then become "never drive, regardless of the state." Intuitively this makes sense, because if the rover can obtain as much energy at the bottom of the hill as it can at the top there is no reason to expend effort trying to stay at (or relocate to) the top of the hill.