CS747 - Assignment 4

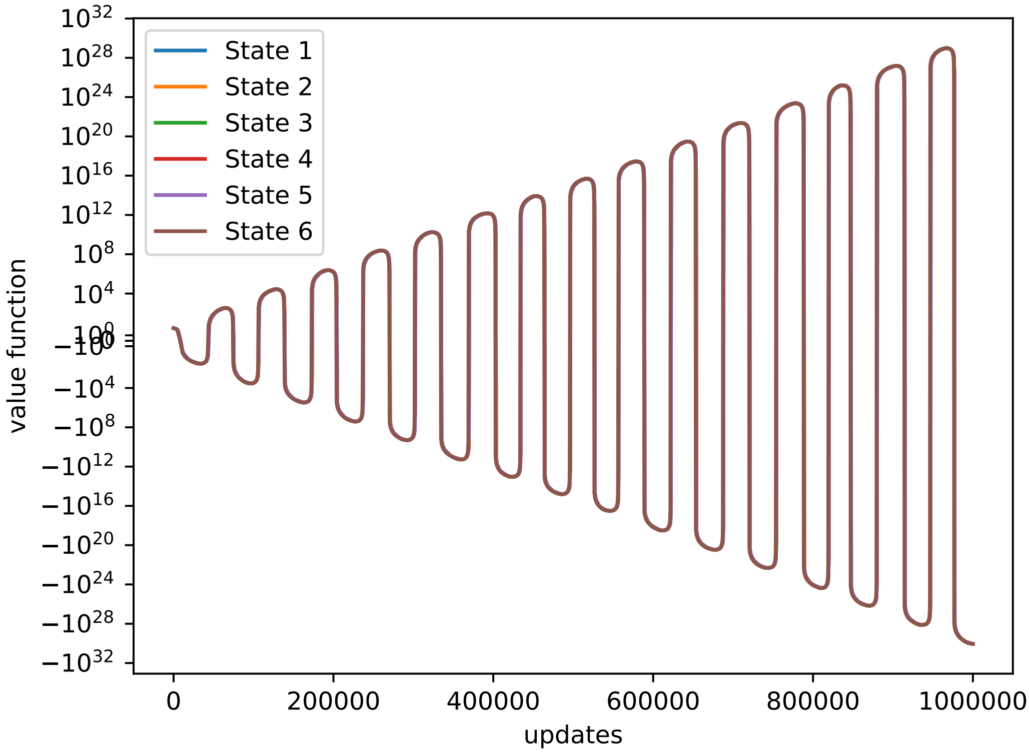
Kalpesh Krishna
140070017
kalpeshk2011@gmail.com

1 Implementation & Results

All algorithms were implemented in Python without the use of any external library except `numpy` to generate random numbers and carry out matrix updates. The `weights` array has been offset by 1 to start indices from 1 (instead of 0) to match Baird's counterexample diagram. A random seed 0 has been used in all experiments.

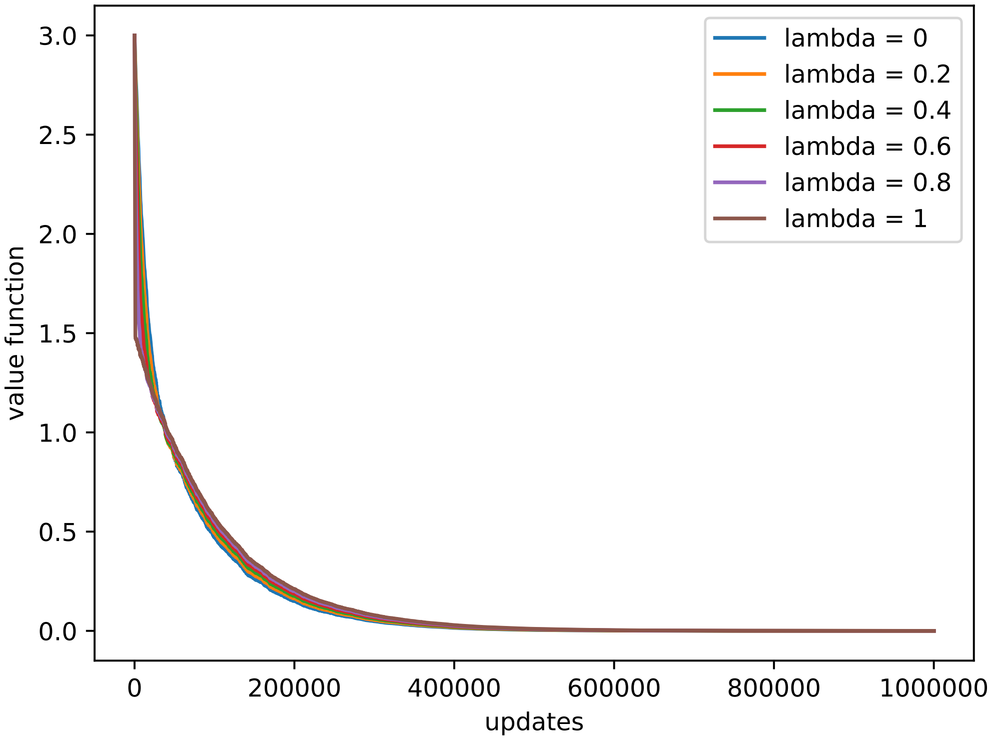
1.1 Experiment 1

In this experiment, all non-terminal states are chosen in a round robin fashion followed by a TD(0) style update. The figure below shows the graph obtained. This is similar in shape to the graph obtained in chapter 8 of Sutton and Barto 1998.
Explanation - The reason for this anomalous behaviour lies in the updates of $w[7]$. Initially, all weights are 1, so $\gamma V[6] < V[1...5]$, which cause $w[1...5], w[7]$ to decrease. $w[6]$ almost never updates due to the 99% self loop. But $w[1...5]$ decrease faster than $w[7]$ and eventually $\gamma V[6] > V[1...5]$. This causes an increase in $V(6)$, until $\gamma V[6] < V[1...5]$, and this cycle continues. Since the value of V(6) has a stronger dependence on $w[7]$ when compared to $V[1...5]$, continuous updates to state 1...5 in response to a strong positive or negative reward due to $\gamma V[6]$ makes $w[7]$ got to positive infinity, and hence $w[1...5]$ go to positive infinity. $w[6]$ on the other hand goes to negative infinity in an attempt to counteract the effect of $w[7]$. But the infrequent updates to $w[6]$ slow this down.



1.2 Experiment 2

No significant difference was observed by changing the value of λ. Unlike the previous case, $V[1...5]$ are updated less often. $w[1...5]$ are updated exactly once for each non-zero update in $w[6]$ in the TD(0) case. In the previous task, $w[6]$ is non-zero updated roughly once in 600 iterations, whereas $w[1...5]$ are updated 100 times each.



1.3 Experiment 3

I noticed that changing the initial set of weights did not affect the final value function achieved. However, some configurations were taking longer to converge, which I suspect is due to a constant learning rate. The final set of weights were different, they differed by a constant factor. However, the ratio of $w[1...5] : w[6] : w[7]$ was always roughly $1 : 4 : -2$, which is expected behavior due to the choice of function approximation. Raw output has been added below.

```
./startmdp.sh 2 1000000 0 1 1 1 1 1 1 1
Weights
[ 0.27999309  0.28000883  0.28002641  0.28001831  0.2799872   1.11999228
 -0.55999853]
Value Function
-1.23541347308e-05 1.91397635615e-05 5.42867724099e-05 3.8093030442e-05 -2.4125545401e-05 -4.7
./startmdp.sh 2 1000000 0 3 1 10 1 10 1 2
Weights
[ 0.99940458  0.99880368  1.00166754  0.9984671   1.00126741  4.00009876
 -1.99999732]
Value Function
-0.0011881549616 -0.00238995587097 0.00333776830031 -0.00306312343532 0.00253750595343 0.00010
./startmdp.sh 2 1000000 0 100 1 0 1 4 80 3
Weights
[ 16.82400073  16.79381982  16.79406201  16.79204024  16.79566909
```

```
  67.20004989 -33.60010428]
Value Function
0.0478971806153 -0.0124646358046 -0.0119802537092 -0.0160238019396 -0.0087660968456 -0.0001586
./startmdp.sh 2 1000000 0 1000 1 0 1 4 80 3
Weights
[  53.04225399   52.73819236   52.74732618   52.72276639   52.74817208
  211.19984362 -105.60095726]
Value Function
0.483550712958 -0.124572552079 -0.106304900622 -0.1554244781 -0.104613113732 -0.00207090789942
./startmdp.sh 2 1000000 0 1000 1 0 1 4 800 3
Weights
[ 168.24305459  167.93828292  167.94312627  167.92048556  167.95005989
  672.00069251 -336.00111036]
Value Function
0.484998817983 -0.124544515748 -0.114857826734 -0.160139244123 -0.100990581897 -0.001528210843
```