CS 4180/5180: Reinforcement Learning and Sequential Decision Making (Fall 2020)– Lawson Wong

Name: [Virender Singh]
Collaborators: [Sunny Shukla]

Question 1.

Response:

(a)

$$V_*(s) = max_a q_*(s,a) \tag{1}$$

(b)

$$q_*(s,a) = \Sigma_{s',r} p(s',r|s,a)[r + \gamma V_*(s',a)] \tag{2}$$

(c)

$$\pi_*(s) = argmax_a q_*(s,a) \tag{3}$$

(d)

$$\pi_*(s) = argmax_a \Sigma_{s',r} p(s',r|s,a)[r + \gamma V_*(s')] \tag{4}$$

(e)

$$v_\pi(s) = \Sigma_a \pi(a|s)[r(s,a) + \gamma \Sigma_{s'} p(s'|s,a) v_\pi(s')] \tag{5}$$

$$v_*(s) = \Sigma_a \pi_*(a|s)[r(s,a) + \gamma \Sigma_{s'} p(s'|s,a) v_*(s')] \tag{6}$$

$$q_\pi(s,a) = [r(s,a) + \gamma \Sigma_{s'} p(s'|s,a) \Sigma_{a'} \pi(a'|s') q_\pi(s',a')] \tag{7}$$

$$q_*(s,a) = [r(s,a) + \gamma \Sigma_{s'} p(s'|s,a) \Sigma_{a'} \pi_*(a'|s') q_*(s',a')] \tag{8}$$

$$\tag{9}$$

Question 2.

Response:

(a)

Algorithm 1: Policy Iteration

Function:

1. Initialization (same as that of textbook)
2. Policy Evaluation (same as that of textbook)
3. Policy Improvement

policy stable $\leftarrow true$

For each s $\in S$:

old action $\leftarrow \pi(s)$

$\pi(s) \leftarrow argmax_a \Sigma_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

"**Change made here"**

if old action $\notin \{\mathbf{a}_i\}$**, where** $q(s, a_i) = q(s, \pi(s))$

or all a_i are equally good as that of $\pi(s) then$,

policy stable $\leftarrow false$

if policy stable, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

(b)

Such a bug does not exist in value iteration because it is off policy i.e. it does not stick to a particular policy and updates value greedily. If two actions produce same value for a state then the Δ will become less than θ and hence the it will exit the loop. There is no policy evaluation step where it might get stuck in the loop.

Question 3.

Response:

(a)

Algorithm 2: Policy Iteration

Function:

1. **Initialization**

V(s) $\in \mathbb{R}$ and $\pi(s) \in \mathbb{A}(s)$ arbitrarily for all s∈S, a∈A

2. **Policy Evaluation**

Loop:

$\Delta \leftarrow 0$

Loop for each s ∈ S and a ∈ A:

q = Q(s,a)

$Q(s,a) \leftarrow \Sigma_{s',r} p(s', r|s, a)[r + \gamma \Sigma_{a'} \pi(a'|s') Q(s', a')]$

$\Delta \leftarrow max(\Delta, |q - Q(s, a)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. **Policy Improvement**

$policystable \leftarrow true$

For each s ∈ S and a ∈ A:

old action $\leftarrow \pi(s)$

$\pi(s) \leftarrow argmax_a Q(s, a)$

If old action $\notin \{a_i\}$ which is the set of equally best solution from $\pi(s)$

Then policy stable $\leftarrow false$

If policy stable then stop and return $Q \approx q_*$ and $\pi \approx \pi_*$; else go to 2

(b)

$$q_{k+1}(s, a) \doteq [R_{t+1} + \gamma max_{a'} q_k(s', a')] \tag{10}$$

$$= \Sigma_{s',r} p(s', r|s, a)[r + \gamma max_{a'} q_k(s', a')] \tag{11}$$

Question 4.

Response:

(a)

The agent at state x would not like to stay at x because it will get -1 reward, if it moves to state y the reward will get -2. Therefore it will try to reach terminal state z which can be done by doing action c and thus the optimal policy would return action c at state x. The agent at state y would not like to stay at y because then it will get -2 reward. If it does action c then it is very highly likely that it will return to the same state. Therefore, if it does action b then it will reach action x, it will get reward of -1 from where it can reach the terminal state. Thus intuitively optimal policy is $\pi_*(x) = c$ and $\pi_*(y) = b$.

(b)

Applying policy iteration algorithm:

1. Initialization:

V(x) = 0, V(y) = 0, $\pi(x) = c, \pi(y) = c$

2. Policy Evaluation:

For each iteration, we have :

Iteration 1 :

for state x:

$$V(x) = p(x,-1|x,c)[-1 + V(x)] + p(y,-2|x,c)[-2 + V(y)] + p(z,0|x,c)[0 + 0]$$
$$V(x) = 0.9(-1 + V(x))$$
$$V(x) = -0.9 + 0.9V(x)$$
$$V(x) = -0.9$$

for state y:

$$V(y) = p(x,-1|y,c)[-1 + V(x)] + p(y,-2|y,c)[-2 + V(y)] + p(z,0|y,c)[0 + 0]$$
$$V(y) = 0.9(-2 + V(y))$$
$$V(y) = -1.8 + 0.9V(y)$$
$$V(y) = -1.8 + 0.9V(y)$$
$$V(y) = -1.8$$

Iteration 2 :

for state x:

$$V(x) = p(x,-1|x,c)[-1 + V(x)] + p(y,-2|x,c)[-2 + V(y)] + p(z,0|x,c)[0 + 0]$$
$$V(x) = 0.9(-1 + V(x))$$

$V(x) = -0.9 + 0.9V(x)$

$V(x) = -1.71$

for state y:

$V(y) = p(x, -1|y, c)[-1 + V(x)] + p(y, -2|y, c)[-2 + V(y)] + p(z, 0|y, c)[0 + 0]$

$V(y) = 0.9(-2 + V(y))$

$V(y) = -1.8 + 0.9V(y)$

$V(y) = -1.8 + 0.9V(y)$

$V(y) = -3.42$

and going in the same way for multiple iteration, we have V(x) = -9 and V(y) = -18,
Since we have the equations for each iterations in the form :

$V(x) = -0.9 + 0.9V(x)$

$V(y) = -1.8 + 0.9V(y)$

We can directly solve these equations. Solving the above equations for states x and y:

$$V(x) = -0.9 + 0.9V(x) \tag{12}$$

$$0.1V(x) = -0.9 \tag{13}$$

$$V(x) = -9 \tag{14}$$

$$\tag{15}$$

$$V(y) = -1.8 + 0.9V(y) \tag{16}$$

$$0.1V(y) = -1.8 \tag{17}$$

$$V(y) = -18 \tag{18}$$

$$\tag{19}$$

3. Policy Improvement:

for state x

old action = c

$\pi(s) = argmax_a\{0.9 \times (-1 + V(x)) + 0.1 \times (0), 0.8 \times (-2 + V(y)) + 0.2 \times (-1 + V(y))\}$

$\pi(x) = argmax_a\{0.9(-1 - 9), 0.8(-2 - 18) + 0.2(-10)\}$

$\pi(x) = argmax_a\{-9, -18\} = c$

for state y

old action = c

$\pi(s) = argmax_a\{0.9 \times (-2 + V(y)) + 0.1 \times (0), 0.8 \times (-1 + V(x)) + 0.2 \times (-2 + V(y))\}$

$\pi(y) = argmax_a\{0.9(-2-18), 0.8(-1-9) + 0.2(-2-18)\}$
$\pi(y) = argmax_a\{-18, -12\} = b$

The optimal policy is $\pi_*(x) = c, \pi_*(y) = b$ and the value of states are V(x) = -9 and V(y) = -18.

(c)

Let's assume the initial policy has action b in both states.

$$V(x) = p(y, -2|x, b)(-2 + V(y)) + p(x, -1|x, b)(-1 + V(x)) \tag{20}$$
$$V(x) = 0.8(-2 + V(y)) + (0.2)(-1 + V(x)) \tag{21}$$
$$V(x) = -1.6 + 0.8V(y) - 0.2 + 0.2V(x) \tag{22}$$

$$V(y) = p(y, -2|y, b)(-2 + V(y)) + p(x, -1|y, b)(-1 + V(x)) \tag{23}$$
$$V(y) = 0.2(-2 + V(y)) + (0.8)(-1 + V(x)) \tag{24}$$
$$V(y) = -1.2 + 0.8V(x) + 0.2V(y) \tag{25}$$

$$0.8V(x) - 0.8V(y) + 1.8 = 0 \tag{26}$$
$$0.8V(x) - 0.8V(y) - 1.2 = 0 \tag{27}$$
$$\tag{28}$$

The equations are inconsistent and hence there is no solution for these pair of equations.
Yes discounting helps because the above equations without discounting don't converge in real values and values tend towards $-\infty$. Yes discounting factor will result in change in optimal policy. Here in this MDP, if we choose a small discounting factor γ the agent at state y will choose action c over action b in the old policy.

Question 5.

Response:

(a)

Value function obtained by running Iterative policy evaluation is shown below:

3.3	8.8	4.4	5.3	1.5
1.5	3	2.3	1.9	0.6
0.1	0.7	0.7	0.4	-0.4
-1	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2

It matches with value function given in figure 3.2 of the textbook.

To run the policy evaluation please run q_a() from main.py.

(b)

Optimal Value Function obtained by running value iteration is shown below:

22	24.4	22	29.4	17.5
19.8	22	19.8	17.8	16
17.8	19.8	17.8	16	14.4
16	17.8	16	14.4	13.3
14.4	16	14.4	13	11.7

Initial policy plot is shown in figure 1 :

Optimal policy plot obtained by running value interation is shown in figure 2:

It matches with v_*, π_* given in figure 3.5 of the textbook.

To run the Value Iteration please run q_b() from main.py.

(c)

Optimal Value Function obtained by running policy iteration is shown below:

22	24.4	22	29.4	17.5
19.8	22	19.8	17.8	16
17.8	19.8	17.8	16	14.4
16	17.8	16	14.4	13.3
14.4	16	14.4	13	11.7

Initial policy plot is shown in figure 3 :

Optimal policy plot obtained by running policy interation with the fix proposed in Q2(a) is shown in figure 4:

It matches with v_*, π_* given in figure 3.5 of the textbook.

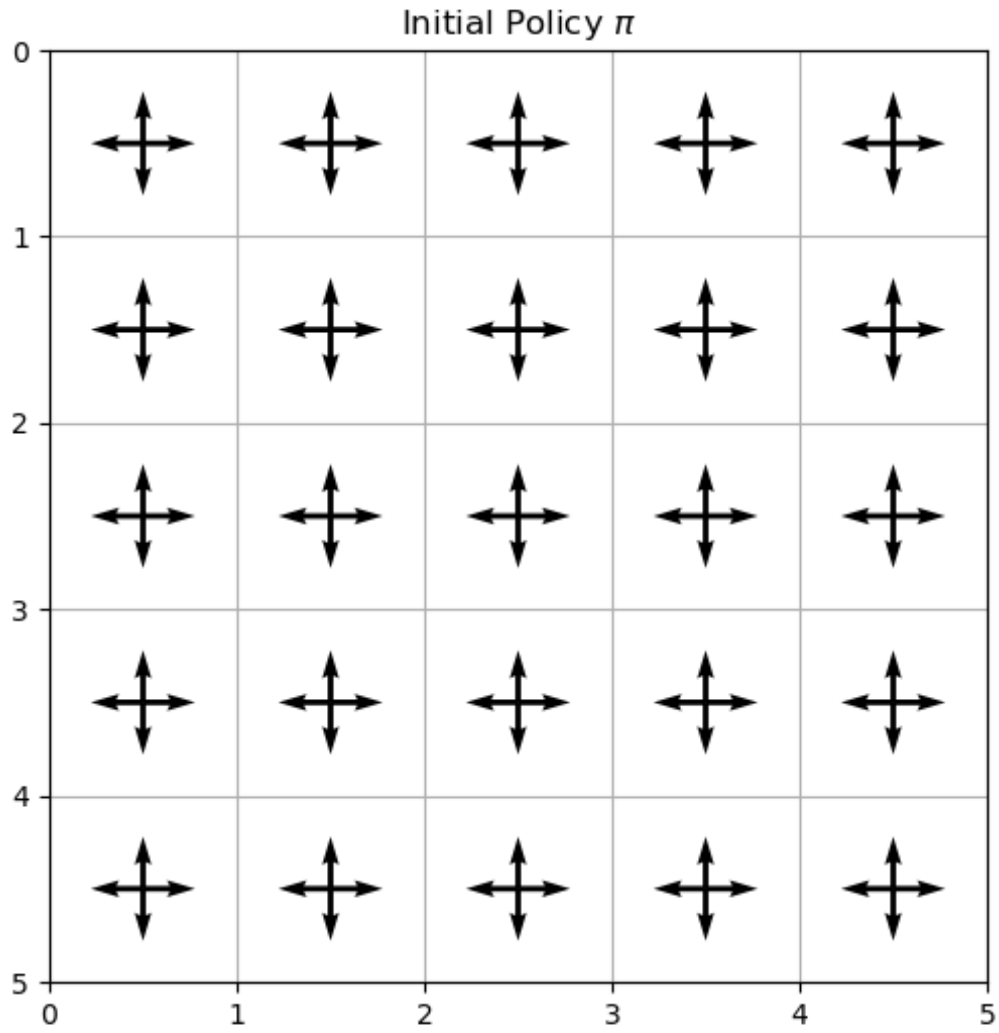

Figure 1: Q5(b) : Initial policy

To run the policy Iteration please run q_c() from main.py. The results are same to the one obtained in part b.

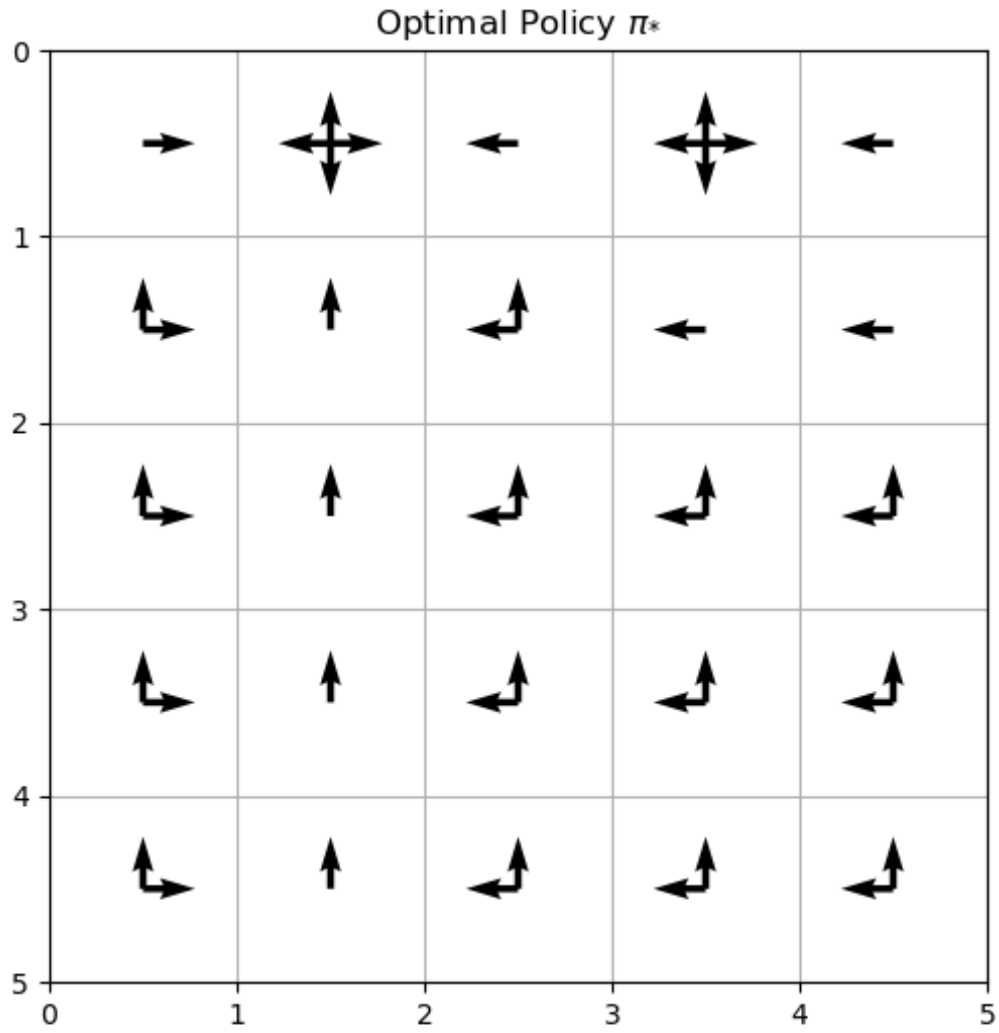

Figure 2: Q5(b) : Optimal Policy after running value iteration

Question 6.

Response:

(a)

We can see in the figure 5 the sequence of policies obtained and the final value function. It matches that of plots shown in figure 4.2. This uses the same policy iteration as implemented in Q5.

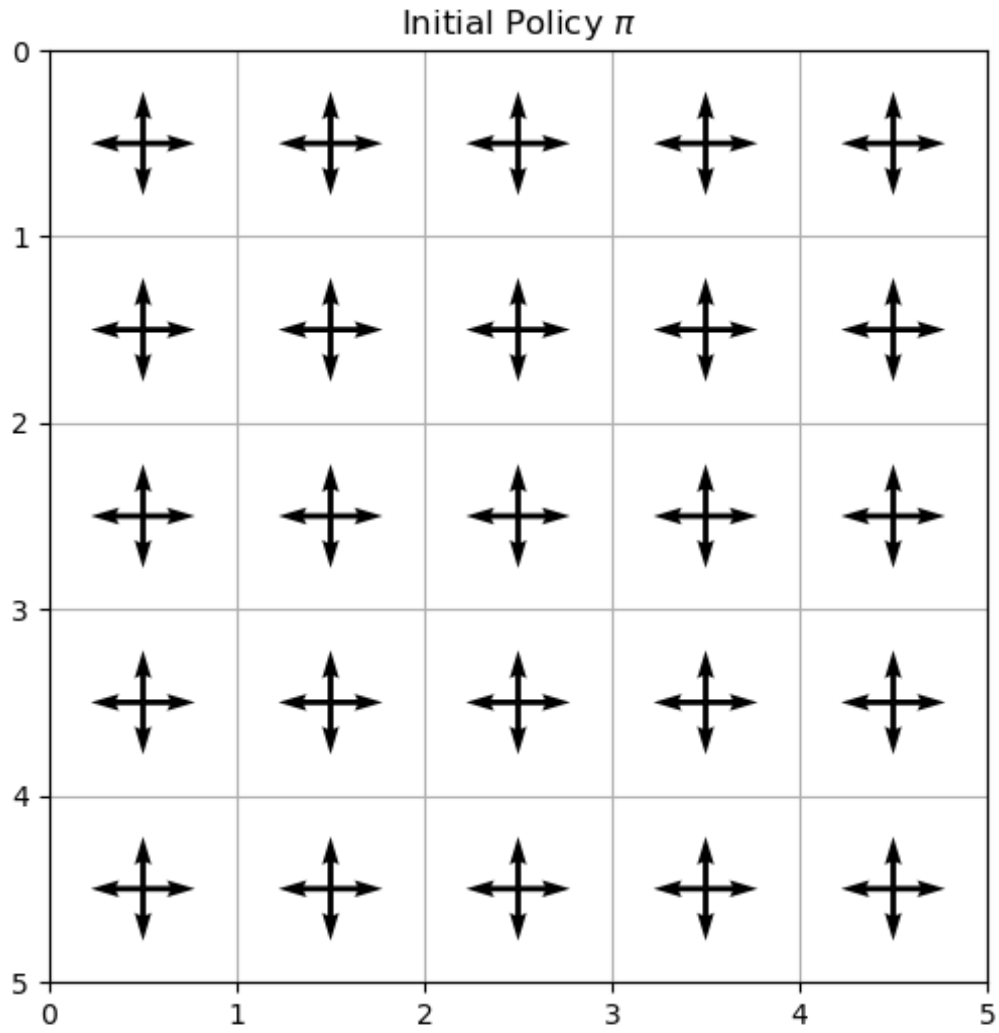

Figure 3: Q5(c) : Initial policy

(b)

Now the reward will change slightly as now we have additional conditions on the cost. For example, we are getting one car shift for free so we would pay 2$ less fine but on the same time if we have more than 10 cars at any location, we will pay 4$ fine. So we will have to incorporate this into the account.

The policy differes from part(a) as in part (a), the optimal policy was not to move as many

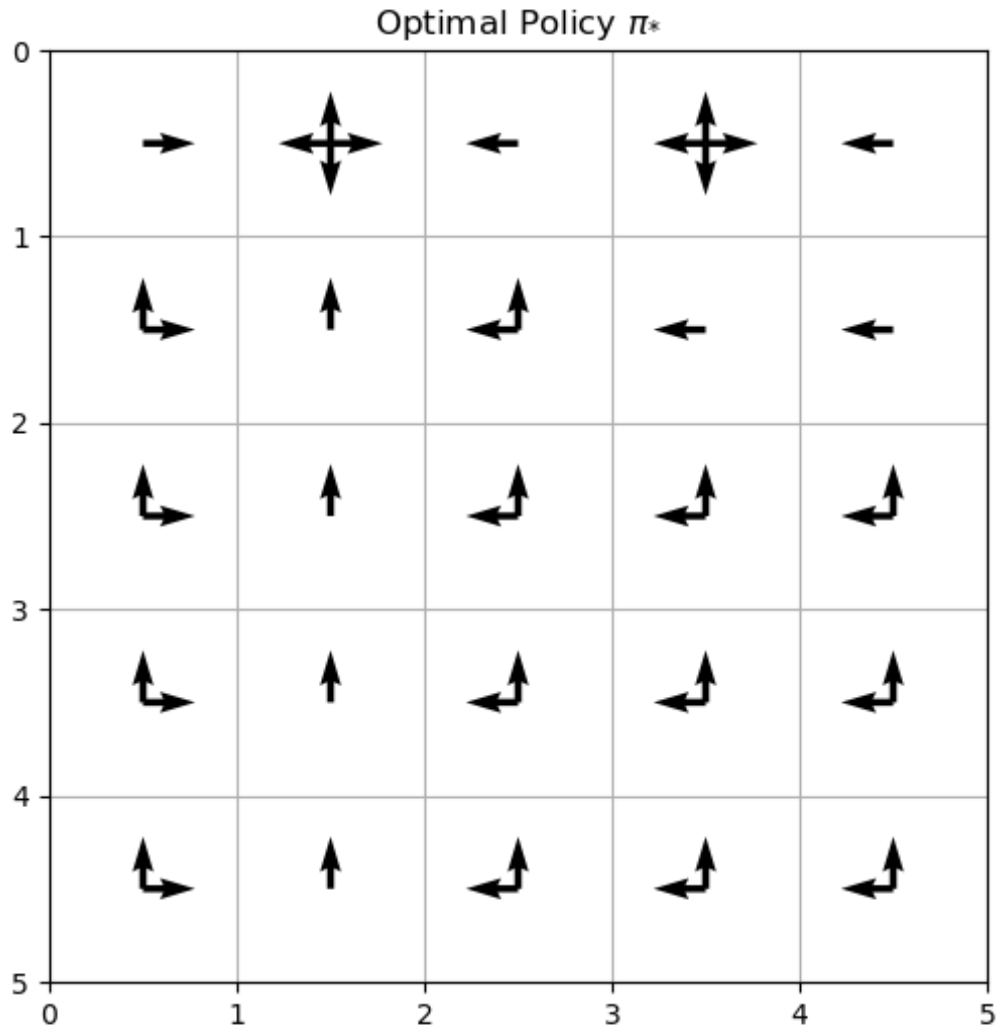

Figure 4: Q5(c) : Optimal Policy after running policy iteration

cars because of shifting fees but now we have one car's shift free and if we don't move, then we have the fine of 4$ per parking location. So optimal policy now is to shift cars regularly to avoid the parking fine and make use of concession in the shifting fee.

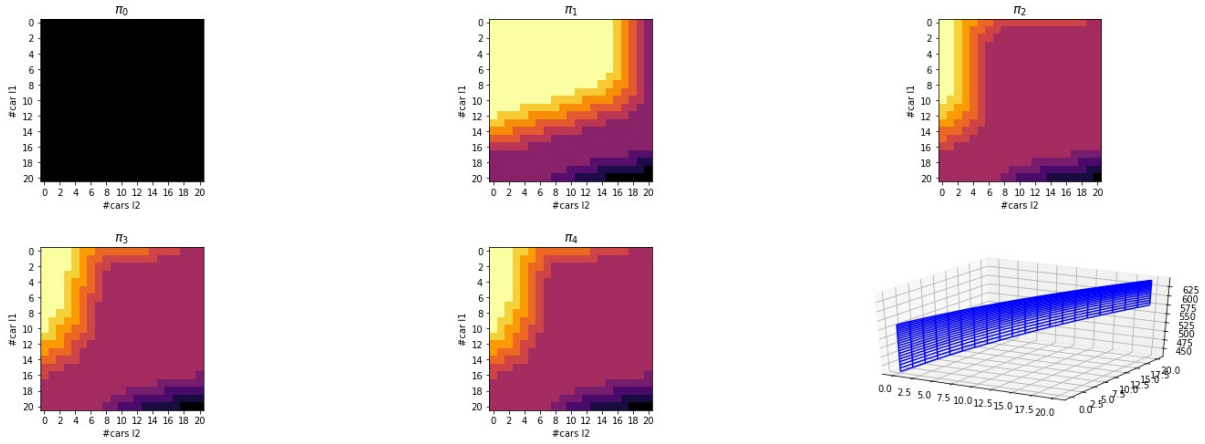

Figure 5: Q6(a) : sequence of policies found by policy iteration on Jack's car rental problem, and the final state-value function

Question 7.

Response:

(a)

To show :

$$|max_a f(a) - max_a g(a)| = max_a f(a) - max_a g(a) \tag{29}$$

Case 1 : $max_a f(a) \geq max_a g(a)$ Let's assume f has its maximum value at a*

$$|max_a f(a) - max_a g(a)| = max_a f(a) - max_a g(a) \tag{30}$$

$$= f(a*) - max_a g(a) \tag{31}$$

$$\leq f(a*) - g(a*) \qquad \text{because g(a*)} \leq max_a \text{ g(a)} \tag{32}$$

$$\leq max_a |f(a) - g(a)| \qquad \text{by definition of max} \tag{33}$$

$$\tag{34}$$

Case 2 : $max_a f(a) < max_a g(a)$ Let's assume g has its maximum value at a*

$$|max_a f(a) - max_a g(a)| = max_a g(a) - max_a f(a) \tag{35}$$
$$= g(a*) - max_a f(a) \tag{36}$$
$$\leq g(a*) - f(a*) \qquad \text{because f(a*)} \leq max_a \text{ f(a)} \tag{37}$$
$$\leq max_a |g(a) - f(a)| \qquad \text{by definition of max} \tag{38}$$
$$\leq max_a |f(a) - g(a)| \qquad \text{by definition of mod} \tag{39}$$
$$\tag{40}$$

(b) To show :

$$\|BV_i - BV_i'\|_\infty \leq \gamma \|V_i - V_i'\|_\infty \tag{41}$$

Proof:

$$\|BV_i(s) - BV_i'(s)\| = \|V_{i+1}(s) - V_{i+1}'(s)\| \tag{42}$$
$$= \|max_a \Sigma_{s',r} p(s', r|s, a)[r + \gamma V_i(s')] - max_a \Sigma_{s',r} p(s', r|s, a)[r + \gamma V_i'(s')]\| \tag{43}$$
$$= \gamma \|max_a \Sigma_{s',r} p(s', r|s, a) V_i(s') - max_a \Sigma_{s',r} p(s', r|s, a) V_i'(s')\| \tag{44}$$
$$\leq \gamma max_a \|\Sigma_{s',r} p(s', r|s, a) V_i(s') - \Sigma_{s',r} p(s', r|s, a) V_i'(s')\| \tag{45}$$
using results from part (a) (46)
let a* be the action with the maximum values, so we can remove max (47)
$$= \gamma \|\Sigma_{s',r} p(s', r|s, a*)(V_i(s') - V_i'(s'))\| \tag{48}$$
$$\tag{49}$$

Inserting into the expression of ∞ norm.

$$\|BV_i(s) - BV_i'(s)\|_\infty = max_s \|B(V_i(s) - V_i'(s))\| \tag{50}$$
$$\leq \gamma max_S \|\Sigma_{s',r} p(s', r|s, a*)(V_i(s') - V_i'(s'))\| \tag{51}$$
$$\leq \gamma max_s \|V_i(s) - V_i'(s)\| = \gamma \|V_i - V_i'\|_\infty \tag{52}$$

(c) The result from part (b) tells us that Bellman operator is a contraction operator by a factor of γ, Let Bx = x for some point x, Assume that $V_{k+1} = x$, for some k, Now by definition $V_{k+2} = BV_{k+1} = Bx = V_{k+1}$, therefore after this point V_k does not really change and hence there

cannot be more than one x because else it would not get any closer when applying contraction method using part (b). Since, the fixed point doesn't move and contraction stops there, it can never reach a second fixed point. Hence the fixed point x is unique.
Now let's define error in the estimate at step k as $||V_k - V*||$, where V* is the optimal value function. Therefore,

$$||V_{k+1} - V*||_\infty = ||BV_k - V*||_\infty \tag{53}$$
$$\leq \gamma ||V_k - V*||_\infty \tag{54}$$
$$\leq \gamma^2 ||V_{k-1} - V*||_\infty \tag{55}$$
$$..... \tag{56}$$
$$\leq \gamma^{k+1} ||V_0 - V*||_\infty \to 0 \tag{57}$$
$$\tag{58}$$

Since, error is reduced by a factor of at least γ on each iteration, therefore error decreases as γ^N after N iterations and hence it converges to the optimal value function. Now we know that $V_{k+1} = x$, Therefore

$$||V_{k+1} - V*||_\infty = 0 \tag{59}$$
$$||x - V*||_\infty = 0 \tag{60}$$
$$x = V* \tag{61}$$

Hence x is the unique fixed point and is equal to the optimal value function V*.