* Exercise 3 : Dynamic Programming *

Discussed with : Vaibhav Saraf

Q.1) a) Give an equation of v_* in terms of q_*

→ $$v_*(s) = \sum_a \pi^*(a|s)\, q_*(s,a)$$

b) equation of q_* in terms of v_* & four argument p

→ $$q^*(s,a) = \sum_{s',r} p(s',r|s,a)\,(r + \gamma v_*(s'))$$

c) equation of π^* in terms of q_*

→ $$\pi_*(s) = \underset{a}{\operatorname{argmax}}\; q_*(s,a)$$

d) equation of π_* in terms of v_* & four argument p

→ $$\pi_*(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s',r|s,a)\,[r + \gamma v_*(s')]$$

e) rewrite 4 bellman equations for (v_π, v_*, q_π, q_*) in terms of 2 args function p & 2 args function r

→ $$v_*(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_*(s')\right]$$

$$v_\pi(s) = \sum_a \pi(a|s) \cdot \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_\pi(s')\right]$$

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left[\sum_{a'} \pi(a'|s')\, q_\pi(s',a')\right]$$

$$q_*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left[\max_{a'} q_*(s',a') \right]$$

(Q.2) Fixing policy iteration.

a)

→ we can fix the bug in policy iteration by every time selecting optimal actions with equal probability.

for example:- If a policy consists of 4 actions

$a = \{$ left, Right, up, down $\}$

at each iteration we will choose optimal policy with equal probability.

if left & up are optimal,

$\therefore \pi(a) = \{0.5, 0, 0.5, 0\}$

∴ Modified pseudocode:-

1. Initialization:-

$V(s) \in \mathbb{R}$ & $\pi(s) \in A(s)$ $\forall s \in S$

2. Policy Evaluation:-

loop :-

$\Delta \leftarrow 0$

loop for each $s \in S$:

$v \leftarrow V(s)$

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

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

3. Policy improvement

policy-stable ← True

For each $s \in S$:

old-action ← $\pi(s)$

$\pi(s) \leftarrow \frac{1}{\text{no. optimal actions}} * \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$

If old-action $\neq \pi(s)$, then policy-stable ← false

If policy-stable, then stop & return $V \approx V_*$ & $\pi \approx \pi_*$; else go to 2.

b) I don't think that value iteration has any bug similar to policy iteration.

When we find optimal value function it should be unique, regardless of how many different optimal policies we have.

(Q.3) Policy iteration for action values

a) Initialization:-

$Q(s,a) \in \mathbb{R}$, $\pi(s) \in A(s)$ arbitrarily $\forall s \in S$

Policy evaluation

Loop:-

$\Delta \leftarrow 0$

Loop for each $s \in S$:-

Loop for each $a \in A$:-

$$Q(s,a) \leftarrow \sum_{s',r} p(s', r \mid s, a) * \left[r + \gamma \sum_{a'} \pi(a' \mid s') \cdot Q(s', a') \right]$$

$$\Delta \leftarrow \max_{a} \left(\Delta , \text{abs}(q(s,a) - Q(s,a)) \right)$$

until $\Delta < \theta$

Policy improvement:-

policy_stable $\leftarrow$ true

For each $s \in S$:-

For each $a \in A$:-

$\pi(s) \leftarrow \arg\max_{a} Q(s,a)$

if old action $\neq \pi(s)$

policy stable $\leftarrow$ false

If policy stable then stop & return $Q \approx q^*$ & $\pi \approx \pi^*$

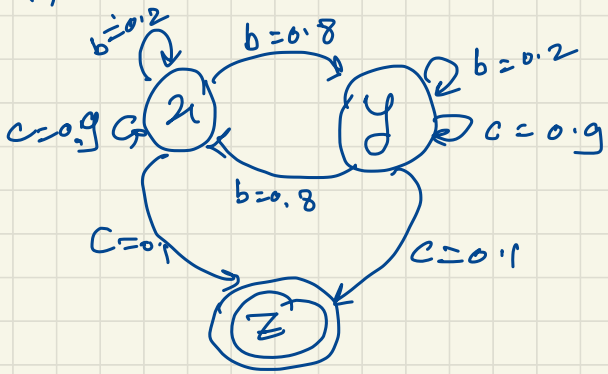
else go to policy evaluation

b)

$$q_{k+1}(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_k(s',a')\right]$$

Q.4) Policy iteration by hand

a)



→ agent's goal is to reach state z (terminal state)

→ But from any state x & y probability of going to terminal state z by taking action 'c' is equal to 0.1.

→ Thus now we will need to look at the rewards

→ on state x agent receives reward '-1' & on state y '-2'

→ To minimize negative rewards agent should stay at x

→ Thus optimal policy could be everytime, from y → move to x → take c from x
from x → take c

b) Iteration 0

$\pi_0 = [x \to c, y \to c]$

$V(x) = r + \gamma \sum_{s'} p(s'|x,c) V(s')$

$= -1 + 0.1 V(z) + 0.9 V(x)$

$V(y) = -2 + 0.1 V(z) + 0.9 V(y)$

$V(z) = 0$

$0.1 V(x) = -1 \quad = -10$

$0.1 V(y) = -2 \quad = -20$

policy improvement

$\pi(x) = \underset{a}{\text{argmax}} \left[r + \gamma \sum p(s'|x,c) V(s') \right]$

$= \underset{a}{\text{argmax}} \begin{cases} -1 + 0.8 * (-20) + 0.2 * (-10), \\ -1 + 0.9 * (-10) \end{cases}$

$= \underset{a}{\text{argmax}} \begin{cases} -1 - 16 - 2 = -19, & \leftarrow b \\ -1 - 9 = -10 & \leftarrow c \end{cases}$

$\pi(x) = c$

$\pi(y) = \underset{a}{\text{argmax}} \begin{cases} -2 + 0.8 * (-10) + 0.2 * (-20) \\ -2 + 0.1 * 0 + 0.9 * (-20) \end{cases},$

$= \underset{a}{\text{argmax}} \{ -14, -20 \} = b$

$\therefore \pi_1 = [x \to c, y \to b]$

Iteration 1

$V(x) = -1 + 0.1 * V(z) + 0.9 * V(x)$

$V(y) = -2 + 0.2 * V(y) + 0.8 * V(x)$

$V(z) = 0$

$\therefore V(x) = -10 \quad V(y) = \frac{-10}{0.8} = -12.5$

Policy improvement:

$$\pi(x) = \underset{a}{\operatorname{argmax}} \begin{cases} -1 + 0.8 * (-12.5) + 0.2 * (-10) \\ -1 + (0.1 * 0) + 0.9 * (-10) \end{cases},$$

$$= \underset{a}{\operatorname{argmax}} \begin{cases} -1 - 10 - 2 \\ -1 - 9 \end{cases},$$

$$= \underset{a}{\operatorname{argmax}} \begin{cases} -13 \leftarrow b \\ -10 \leftarrow c \end{cases}$$

$\pi(x) = c$

$$\pi(y) = \underset{a}{\operatorname{argmax}} \begin{cases} -2 + 0.8 * (-10) + 0.2 * (-12.5) \\ -2 + 0.9 * (-12.5) \end{cases},$$

$$\pi(y) = \underset{a}{\operatorname{argmax}} \begin{cases} -12.5 \leftarrow b \\ -13.25 \leftarrow c \end{cases}$$

$\pi(y) = b$

$\therefore \pi = [x \to c, y \to b]$

Which is same as previous policy. Hence policy is converged.

c) Initial policy

$\pi = [x \to b, y \to b]$

$$V(x) = -1 + 0.8\,V(y) + 0.2\,V(x)$$

$$V(y) = -2 + 0.8\,V(x) + 0.2\,V(y)$$

$$V(z) = 0$$

$$0.8\,V(x) = -1 + 0.8\,V(y)$$

$$0.8\,V(y) = -2 + 0.8\,V(x)$$

$$0.8\,V(x) = -1 + [-2 + 0.8\,V(x)]$$

$$0.8\,V(x) = -3 + 0.8\,V(x)$$

$0 = 3$ which is inconsistent.

Our initial policy always takes action b which causes it to loop between state x & y infinitely.

Since we are considering undiscounted MDP we will get sum of infinite negative numbers.

Does discounting help?

→ Discounting factor can help with above said problem as values in future will become insignificant

→ But ensuring accurate optimal solution will be difficult in this case as the overall problem changes.

→ In this MDP optimal policy is not dependent on discounting factor, as the policy will still be able to converge even with discount rate.

Q.7) Proving convergence of value iteration

for any function f & g

a) $\left|\max_a f(a) - \max_a g(a)\right| \leq \max_a |f(a) - g(a)|$

→ case 1

$$\max_a f(a) - \max_a g(a) \geq 0 \quad -(1)$$

let $\max_a f(a) = a_f^*$

$\max_a g(a) = a_g^*$

$$\max_a f(a) - \max_a g(a) = f(a_f^*) - g(a_g^*) \geq 0 \quad -(2)$$

$$\therefore \left|f(a_f^*) - g(a_g^*)\right| \leq \max_a \left|f(a) - g(a)\right|$$

from (2) we can say that

$g(a^*_f) \leq g(a^*_g)$ as a^*_g is optimal action for g.

Thus we can write inequality from ① as.

$$f(a^*_f) - g(a^*_f) \geq f(a^*_f) - g(a^*_g) \geq 0$$

$$\therefore f(a^*_f) - g(a^*_g) = |f(a^*_f) - g(a^*_f)|$$

$$\therefore f(a^*_f) - g(a^*_g) \leq f(a^*_f) - g(a^*_f)$$

$$= |f(a^*_f) - g(a^*_f)| \leq \max_a |f(a) - g(a)|$$

Case 2:-

$$\max_a f(a) - \max_a g(a) = f(a^*_f) - g(a^*_g) < 0$$

$$f(a^*_g) \leq f(a^*_f)$$

Considering above

$$g(a^*_g) - f(a^*_g) \geq g(a^*_g) - f(a^*_f) > 0$$

as function $g >$ func f

$$g(a^*_g) - f(a^*_g) = |g(a^*_g) - f(a^*_g)|$$

$$\therefore g(a^*_g) - f(a^*_f) \leq g(a^*_g) - f(a^*_g)$$

$$= |g(a^*_g) - f(a^*_g)| \leq \max_a |g(a) - f(a)|$$

$$= \max_a |f(a) - g(a)|$$

Q.7) b)

From Q7 part a)

$$\left| \max_a f(a) - \max_a g(a) \right| \leq \max_a |f(a) - g(a)|$$

$V_k \leftarrow k$th value iteration

$B \leftarrow$ bellman backup operator.

$$V_{k+1} \leftarrow BV_k$$

$$V_{k+1}(s) \leftarrow \max_a \sum P(s', r | s, a) [r + \gamma V_k(s')]$$

$\|V\|_\infty$ is max absolute value of its elements

→

from definition

$$\|BV_1 - BV_1'\|_\infty = \|R(s) + \max_a \gamma \sum P(s'|s,a) V(s') - (R(s) - \max_a \gamma \sum P(s'|s,a) V'(s')\|_\infty$$

$$= \|\max_a \gamma \sum P(s'|s,a) \cdot V(s') - \max_a \gamma \sum P(s'|s,a) V'(s')\|_\infty$$

= from previous eqn. above eqn. is less than or equal to,

$$\leq \max_a |\gamma \sum P(s'|s,a) V(s') - \gamma \sum P(s'|s,a) V'(s')|$$

$$\leq \max_a \gamma |\sum P(s'|s,a) \cdot V(s') - \sum P(s'|s,a) \cdot V(s')|$$

$$\leq \max_a \gamma \sum P(s'|s,a) \|V - V'\|$$

$$= \max_a \gamma \|V - V'\|$$

we know that sum of probabilities s' given a state & action to take

$$\sum_{s'} P(s'|s,a) = 1$$

$$\therefore \max_a \gamma \|V - V'\| = \gamma \|V - V'\|$$

$$\therefore \|BV_i - BV_i'\|_\infty \leq \gamma \|V_i - V_i''\|_\infty$$

Question 5 Answer

Part A

```
Value Function for iterative policy evaluation
[-1.9 -1.3 -1.2 -1.4 -2.  -1.  -0.4 -0.3 -0.6 -1.2  0.1  0.7  0.7  0.4
 -0.4  1.5  3.   2.3  1.9  0.6  3.3  8.8  4.4  5.3  1.5]
```

Part B

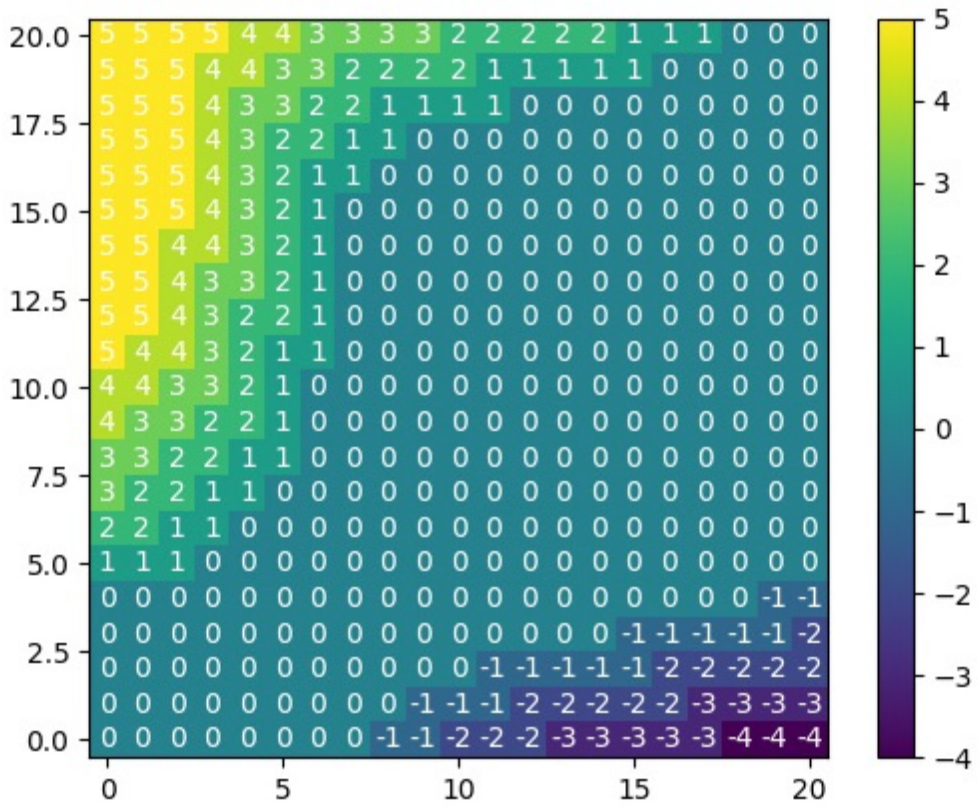
```
Optimal Value function for Value Iteration
[14.4 16.  14.4 13.  11.7 16.  17.8 16.  14.4 13.  17.8 19.8 17.8 16.
 14.4 19.8 22.  19.8 17.8 16.  22.  24.4 22.  19.4 17.5]
```

```
("Optimal policy for Value Iterationdefaultdict(<class 'list'>, {(0, 0): "
 "['RIGHT', 'UP'], (0, 1): ['UP'], (0, 2): ['LEFT', 'UP'], (0, 3): ['LEFT', "
 "'UP'], (0, 4): ['LEFT', 'UP'], (1, 0): ['RIGHT', 'UP'], (1, 1): ['UP'], (1, "
 "2): ['LEFT', 'UP'], (1, 3): ['LEFT', 'UP'], (1, 4): ['LEFT', 'UP'], (2, 0): "
 "['RIGHT', 'UP'], (2, 1): ['UP'], (2, 2): ['LEFT', 'UP'], (2, 3): ['LEFT', "
 "'UP'], (2, 4): ['LEFT', 'UP'], (3, 0): ['RIGHT', 'UP'], (3, 1): ['UP'], (3, "
 "2): ['LEFT', 'UP'], (3, 3): ['LEFT'], (3, 4): ['LEFT'], (4, 0): ['RIGHT'], "
 "(4, 1): ['LEFT', 'DOWN', 'RIGHT', 'UP'], (4, 2): ['LEFT'], (4, 3): ['LEFT', "
 "'DOWN', 'RIGHT', 'UP'], (4, 4): ['LEFT']})")
```

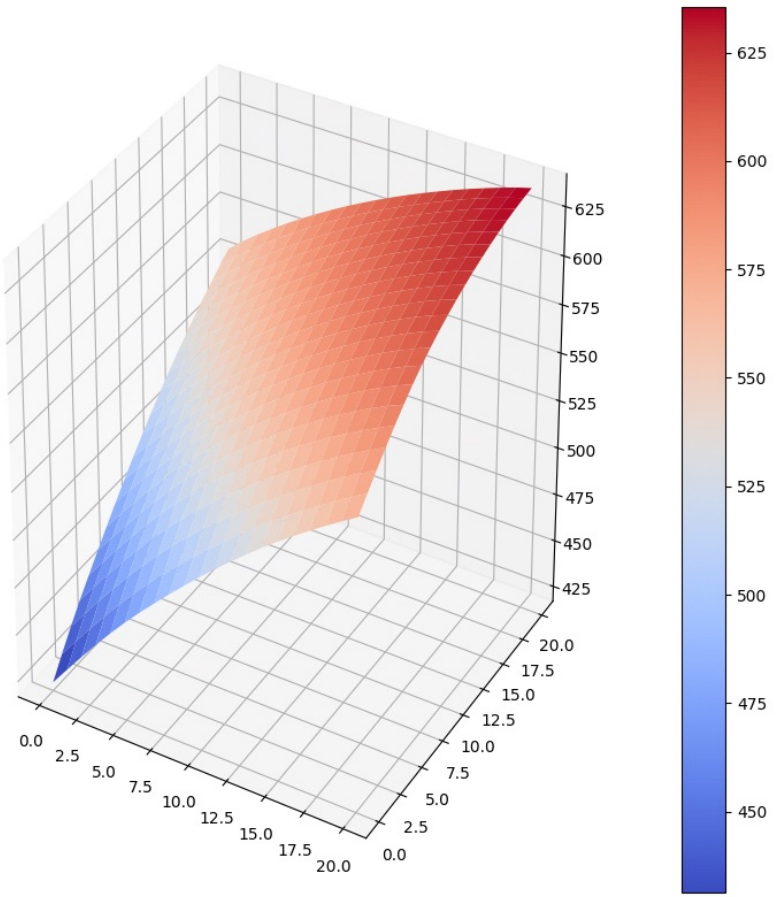
Part C sequence of actions in array is [LEFT,DOWN,RIGHT,UP]

```
Optimal Value function for Policy Iteration
[14.4 16.  14.4 13.  11.7 16.  17.8 16.  14.4 13.  17.8 19.8 17.8 16.
 14.4 19.8 22.  19.8 17.8 16.  22.  24.4 22.  19.4 17.5]
('Optimal policy function for Policy Iteration \n'
 ' [[0.   0.   0.5  0.5 ]\n'
 ' [0.   0.   0.   1.  ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.   0.   0.5  0.5 ]\n'
 ' [0.   0.   0.   1.  ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.   0.   0.5  0.5 ]\n'
 ' [0.   0.   0.   1.  ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [0.   0.   0.5  0.5 ]\n'
 ' [0.   0.   0.   1.  ]\n'
 ' [0.5  0.   0.   0.5 ]\n'
 ' [1.   0.   0.   0.  ]\n'
 ' [1.   0.   0.   0.  ]\n'
 ' [0.   0.   1.   0.  ]\n'
 ' [0.25 0.25 0.25 0.25]\n'
 ' [1.   0.   0.   0.  ]\n'
 ' [0.25 0.25 0.25 0.25]\n'
 ' [1.   0.   0.   0.  ]]')
```

Question 6 Part A Optimal Policy

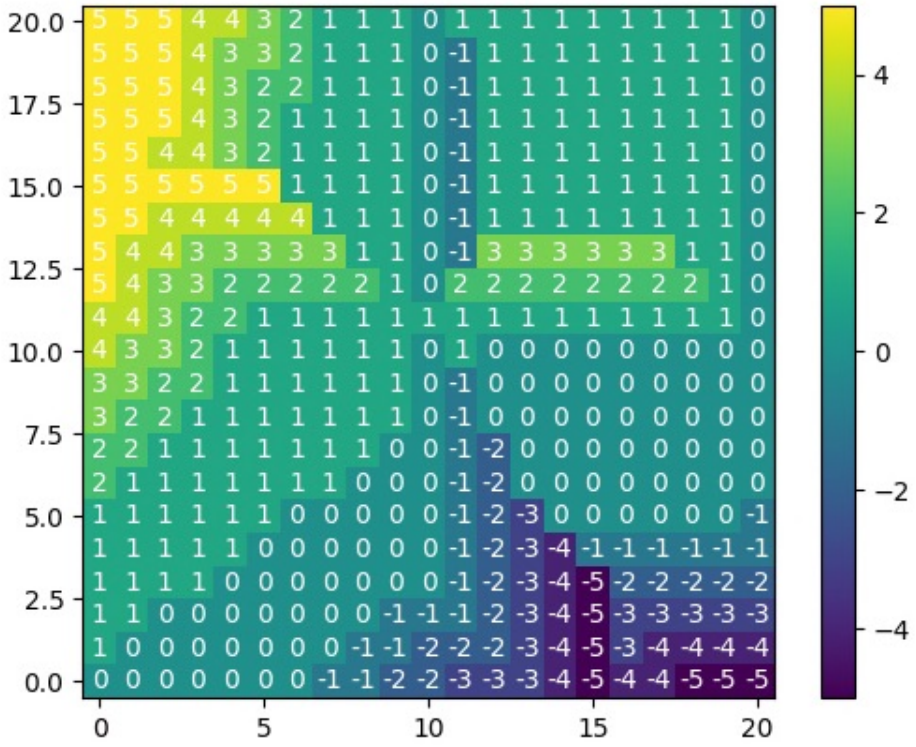


Question 6 A Optimal Value function

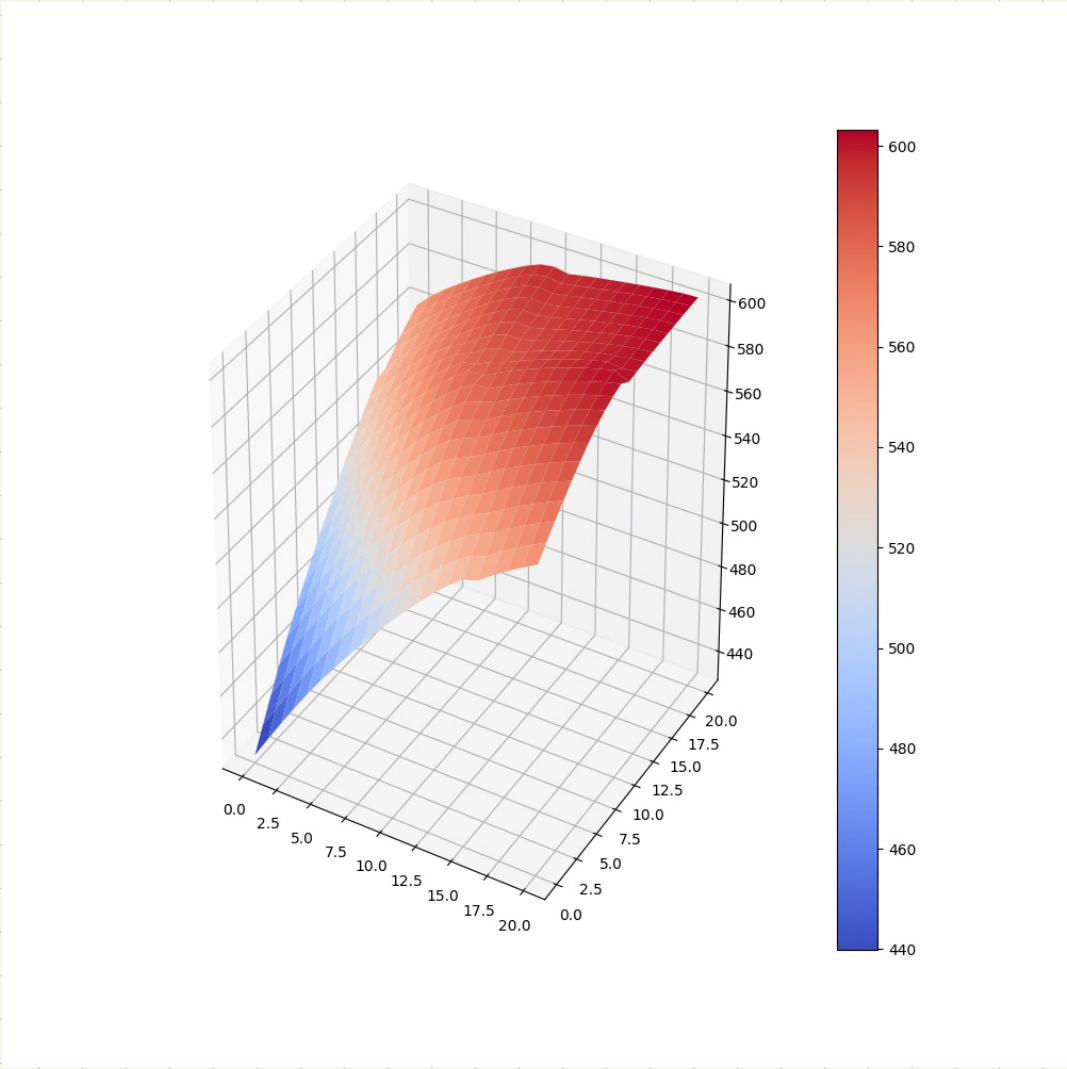


Question 6 Part B (Modified Jacks Car Rental)

Optimal Policy



Optimal Value Function for Modified Jack’s Car Rental



written: How do we change dynamics function to reflect modified Jack's car rental problem

Answer:
We mainly change the way we calculate reward to ensure we include cases with changed costs. These changes are mainly in overnight cost to move cars. We check if the action is positive, i.e. moving car from A to B, then our cost is reduced by 2$ since we get 1 free car move. Also, we check if at any location we have more than 10 cars. Thus, we place additional cost of 4$.

Written: How does final policy differ from original Jack's Car Rental Problem

Ans :
One immediate observation can be done in optimal policy of jack's car rental is, it tries to move atleast 1 car from location 1 to 2 since its free to move except when location 2 has exactly 10 cars, since we'll need to pay overflow parking cost