Course Code: CS 5180

Course: Reinforcement Learning and Sequential Decision Making

Name: Pavan Rathnakar Shetty

Please find the entire submission in Canvas and https://github.com/Pavan-r-shetty/Reinforcement-Learning-2023.git as well

EX3 Assignment Submission

Ex 3

1) a) $v_\pi(s)$: Expected return starting from state s and following policy π

$v_*(s)$: Maximum expected return starting from state s

$q_\pi(s,a)$: Expected return taking action a in state s and the following policy π

p: Transition Probabilities

r: Expected rewards

$v_*(s) = \max_a q^*(s,a)$

b) $q_* = r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_*(s')$

c) $\pi_*(s) = \arg\max_a q_*(s,a)$

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

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

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

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

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

3)

a) 1. Random Initialization of $Q(s,a)$ and $\pi(s)$

2. Policy Evaluation

loop: $\Delta \leftarrow 0$

for $s \in S$

loop:

for $a \in A$

$$Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)\left(r + \gamma Q(s', \pi(s'))\right)$$

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

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

3. Policy Improvement

Policy stable $\leftarrow$ true

for each $s \in S$:

old-action $\leftarrow \pi(s)$

$$\pi(s) \leftarrow \arg\max_a Q(s,a)$$

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

If Policy-stable, then stop and return $V \approx v$ and $\pi \approx \pi$ else go to 2

b) The Analog of value iteration update equation for action values will be,

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

For value iteration, we perform simultaneous updates to our value functions, without distinguishing between policy evaluation and policy improvement phases.

The equation provides the action value for a specific action a taken in state s by considering the immediate reward and the discounted value of the expected next state-action pair.

2) a) Finite Policy Iteration:

The problem with the policy iteration algorithm is that it can loop forever if there are multiple policies that have the same value but are different in terms of the actions they specify. The algorithm keeps alternating between these equivalent policies.

To guarantee convergence, we can introduce a termination condition that checks if the value function remains unchanged (or changes less than a certain small threshold) after a policy improvement step.

Modify Policy Iteration:

1. Initialization:

$\pi(s)$ arbitrarily for all s

$V(s)$ arbitrarily for all s

2. Policy Evaluation:

Iterate:

$V(s) = \sum_a \pi(a|s) \left(R_s^a + \gamma_{s'} P_{ss'}^a V(s') \right)$

until $V(s)$ converges

3. Policy Improvement:

policy_changed = False

For each s:

$a^* = \arg\max_a \left(R_s^a + \gamma \sum_{s'} P_{ss'}^a V(s') \right)$

if $a^* \neq \pi(s)$

$\pi(s) = a^*$

policy_changed = True

4. Termination:

If not policy_changed, then terminate; otherwise, go back to Policy Evaluation step.

In this modification, the algorithm will only loop back to the policy evaluation step if the policy has actually changed. This ensures that the algorithm will converge even if there are multiple optimal policies.

b) No, there isn't an analogous bug in the value iteration. Value iteration is not based on the two-step process of policy evaluation followed by policy improvement. Instead, it directly updates the value function by taking the maximum overall actions.

Value iteration guarantees that the value function will converge to V^*, the optimal value function, after a sufficient number of iterations. Each iteration brings the value function closer to V^*, so there's no chance of endlessly oscillating between multiple non-optimal value functions.

4)a) Qualitative Analysis:
By examining the transition and reward models, we can make the following observations:
1. Taking any action in state x always results in -1 reward
2. Taking any action in state y always results in -2 reward
3. Taking c action has a 0.15 chance to take the agent to a terminal state, ending the decision-making process.

Given the agent aims to maximize reward (or minimise losses in this case), in state y taking action c is a good choice because there is a chance to move to the terminal state and avoid the -2 reward. For state x, the agent will consider both the immediate reward and the potential future rewards.

b) Policy iteration:

Assuming the initial policy π has action c in both states.

1. Initialization:

$\pi(x) = c$
$\pi(y) = c$

$V(x) = 0$
$V(y) = 0$ (arbitrary)

2. Policy evaluation:

Using the Bellman equation for policy evaluation, we'll update the value of each state under the current policy:

For state x:

$V(x) = -1 + 0.85 \times V(x) + 0.15 \times 0$ (since x is terminal)
$V(x) = -6.67$

For state y:

$V(y) = -2 + 0.85 \times V(y) + 0.15 \times 0$
$= -13.33$

3. Policy Improvement:

For state x:

$Q(x, b) = -1 + 0.2(V_x) + 0.8(V_y) = -12.998$
$Q(x, c) = -1 + 0.85(V_x) = -6.67$

Since $-6.67 > -12$, $\pi(x)$ remains c

For state y:

$Q(y,b) = -2 + 0.8 \times (v_x) + 0.2(v_y) = -7.33$

$Q(y,c) = -2 + 0.95 (v_y) = -13.33$

~~Therefore, $\pi(x)$ remains c as it's the better action~~

Since $-7.33 > -13.33$, $\pi(y)$ switches to b.

after on policy improvement, we have:

$\pi(x) = c$

$\pi(y) = b$

Given the updated policy, we again proceed with policy evaluation to get the new state values:

Policy Evaluation:

For state x with action c (unchanged policy)

$V(x) = -6.67$

For state y with action b

$V(y) = -2 + 0.8 (v_x) + 0.2(v_y)$

$= -7.33$

Policy Improvement: For state x:

$Q(x,b) = -12$

$Q(x,c) = -6.67$

So $\pi(x)$ remains c

For state y:

$Q(y,b) = -7.33$

$Q(y,c) = -13.33$

So, $\pi(y)$ remains b

Now, since policy has not changed after this iteration, policy iteration will terminate.

∴ Optimal Policy:

$\pi^*(x) = c$
$\pi^*(y) = b$

c) $\pi(x) = b$
$\pi(y) = b$

Policy Evaluation:

For state x with action b

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

For state y with action b

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

$V(x) = -15$
$V(y) = -13.75$

Policy Improvement:

For state x:

Following action b:

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

$= -1 + 0.8(-13.75) + 0.2(-15)$

$= -15$

Following action c:

$Q(x,c) = -1$ (terminal) $= -1$

For state y:

Following action b:

$$Q(y,b) = -2 + 0.8(V_x) + 0.2(V_y)$$

$$= -2 + 0.8(-15) + 0.2(-13.75)$$

$$= -16.75$$

Following action c:

$$Q(y,c) = 2$$

Clearly, for both states x & y, the action c results in higher Q values. Thus c is the improved policy.

Does discounting help?

Discounting can indeed influence the optimal policy in some MDPs. In this context, if discount factor γ is included and is very close to 0, it would mean the agent cares very little about the future rewards. The action which gives the highest immediate reward would be favored.

Does the optimal policy depend on the discount factor in this MDP?

Yes, if γ is very close to 0, immediate rewards become more significant and optimal policy may change based on the immediate rewards for each action. However, in this MDP, since the rewards are all negative and state z is a terminal state, the optimum policy will likely remain to transition to z state using action c as it incurs the least negative reward.

7]

a) case 1:

Suppose $\max_a f(a) - \max_a g(a) \geq 0$

$\Rightarrow \max_a f(a) \geq \max_a g(a)$

$\Rightarrow \max_a f(a) - g(a) \geq \max_a f(a) - \max_a g(a)$

The term $\max_a f(a) - g(a)$ is less than or equal to $\max_a |f(a) - g(a)|$. Hence, our inequality holds for this case.

Case 2: Suppose $\max_a f(a) - \max_a g(a) < 0$

$\Rightarrow \max_a f(a) < \max_a g(a)$

$\Rightarrow \max_a g(a) - f(a) > 0$

and

$\max_a g(a) - f(a) \leq \max |f(a) - g(a)|$

Rearranging gives us the required inequality.

b) For any state s:

$$B V_i(s) - B V_i'(s) = \max_a \sum_{s',r} p(s',r|s,a)[\gamma V_i(s') - \gamma V_i'(s')]$$

The absolute difference is:

$$|B V_i(s) - B V_i'(s)| \leq \gamma \max_a \sum_{s',r} p(s',r|s,a) |V_i(s') - V_i'(s')|$$

using part a)

$$\gamma \max_a \sum_{s',r} p(s',r|s,a) |V_i(s') - V_i'(s')| \leq \gamma \|V_i - V_i'\|_\infty$$

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

c) Convergence to Fixed Point:

Given that the Bellman backup operator is a contraction by a factor of γ, and using the result of part (b), the difference between successive iterates $\|V_{i+1} - V_i\|_\infty$ becomes arbitrarily small, which means the value iteration is converging.

Uniqueness of Fixed Point: If there were two distinct fixed points, say x and y, then $\|Bx - By\|_\infty \leq \gamma \|x - y\|_\infty$. However, since $Bx = x$ and $By = y$, this would mean LHS $= 0$, which implies $x = y$, a contradiction. Therefore, the fixed point is unique.

Fixed Point is v^*: For any policy π and any state s, the value function defined by the Bellman operation is the maximum over all actions. Since v^* satisfies this and given

M T W T F S S
Page No.:
Date:
YOUVA

that the fixed point is unique, the fixed point of the Bellman backup operator must be v^*.

Hence, value iteration converges to the unique optimal value function v^*.

5)
Please run the env.py file

a)
Please uncomment the block marked 5a
line 416-422
Please comment blocks marked 5b

line 424-434

Please comment blocks marked 5c

Line 436-500

Ans
[[3.31359559 8.79292942 4.43113177 5.32556099 1.4955287]

[1.52582318 2.99591435 2.2534199   1.91064941 0.55045095]

[0.05486787 0.74165922 0.67626363 0.36114423 -0.40025498]

[-0.96965064 -0.43208514 -0.35180898 -0.58272448 -1.18027658]

[-1.85380443 -1.34185832 -1.22622928 -1.42007309 -1.97241846]]

b)
Please comment the block marked 5a
line 416-422
Please uncomment blocks marked 5b

line 424-434

Please comment blocks marked 5c

Line 436-500

Ans

Optimal Value Function:

[[21.9773651 24.41934924 21.97741432 19.41934924 17.47741432]

[19.77962859 21.97741432 19.77967288 17.8017056 16.02153504]

[17.80166573 19.77967288 17.8017056 16.02153504 14.41938153]

[16.02149916 17.8017056 16.02153504 14.41938153 12.97744338]

[14.41934924 16.02153504 14.41938153 12.97744338 11.67969904]]

Optimal Policy:

[[3 0 1 0 1]

[3 0 0 1 1]

[3 0 0 0 0]

[3 0 0 0 0]

[3 0 0 0 0]]

c)

Please comment the block marked 5a
line 416-422
Please comment blocks marked 5b

line 424-434

Please uncomment blocks marked 5c

Line 436-500

Ans

21.98 24.42 21.98 19.42 17.48

19.78 21.98 19.78 17.80 16.02

17.80 19.78 17.80 16.02 14.42

16.02 17.80 16.02 14.42 12.98

14.42 16.02 14.42 12.98 11.68

Optimal Policy:

Action.UP Action.LEFT Action.DOWN Action.LEFT Action.DOWN

Action.UP Action.LEFT Action.LEFT Action.DOWN Action.DOWN

Action.UP Action.LEFT Action.LEFT Action.LEFT Action.LEFT

Action.UP Action.LEFT Action.LEFT Action.LEFT Action.LEFT

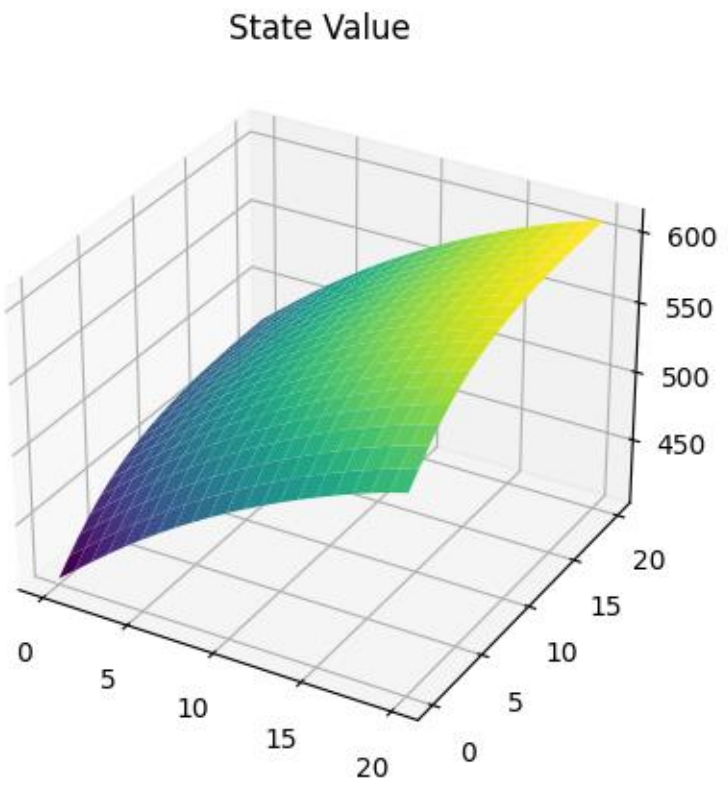
Action.UP Action.LEFT Action.LEFT Action.LEFT Action.LEFT

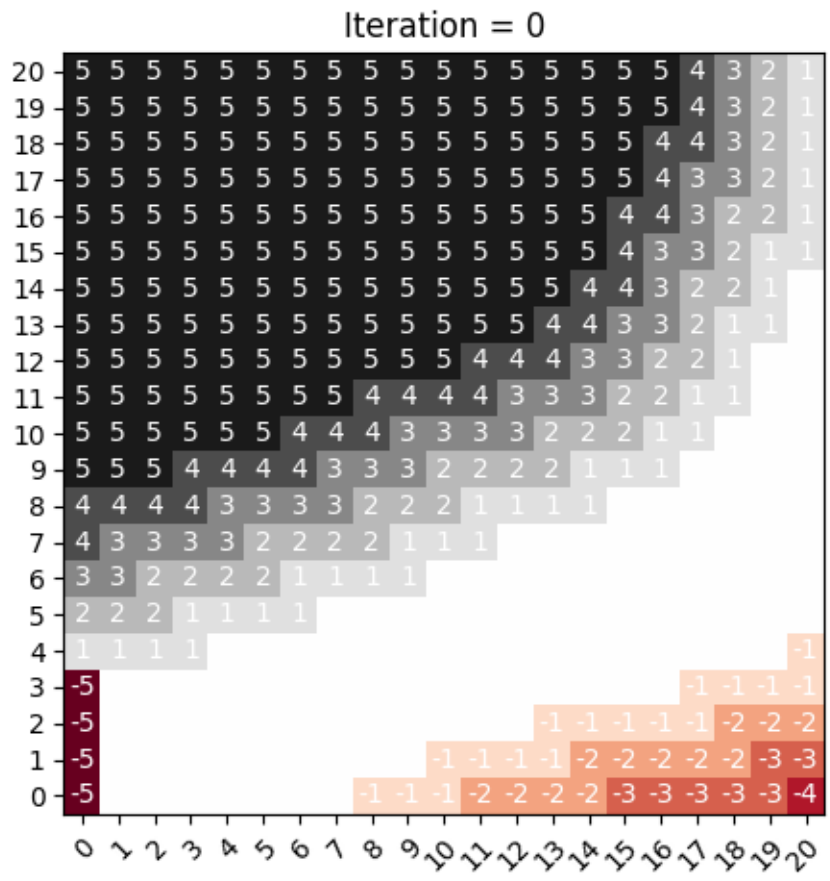
6)

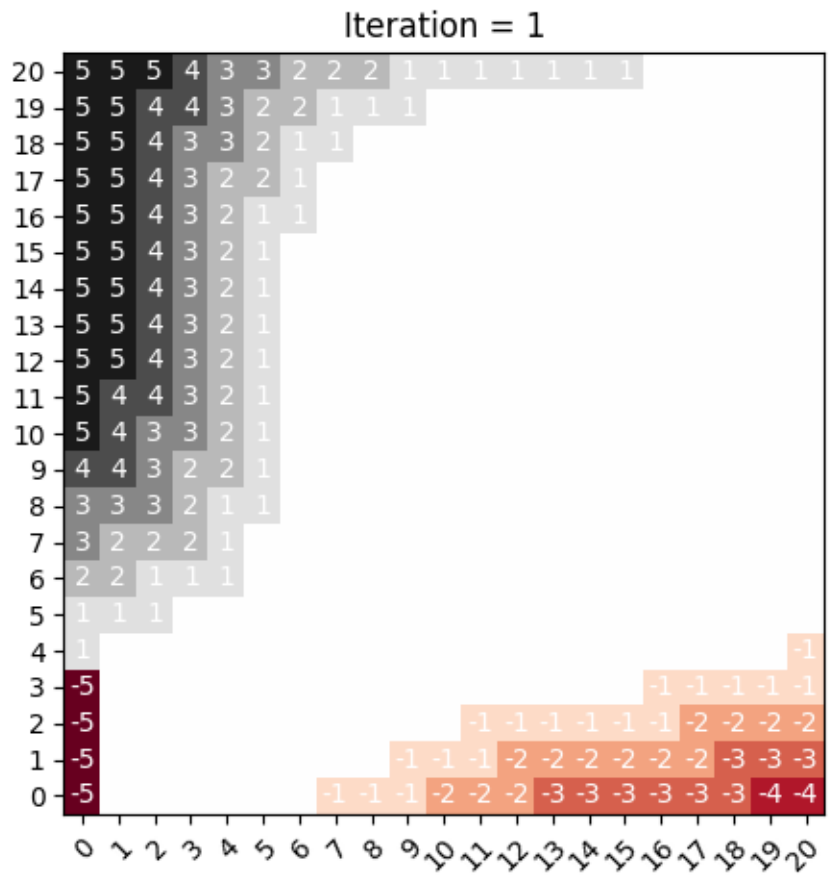
I tried my best to code in env.py using the helper function provided but failed to debug few stuff
Therefore, I wrote a new script that solves Jack's Car Rental Problem in a file named **env.ipynb**

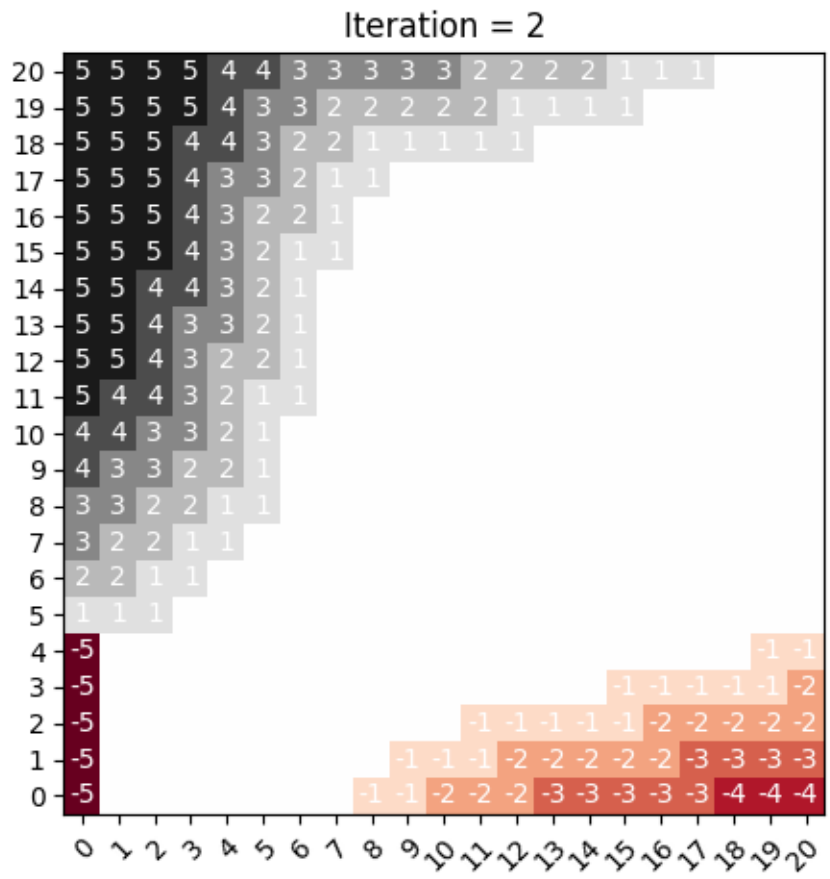
Please use env.ipynb for Q6

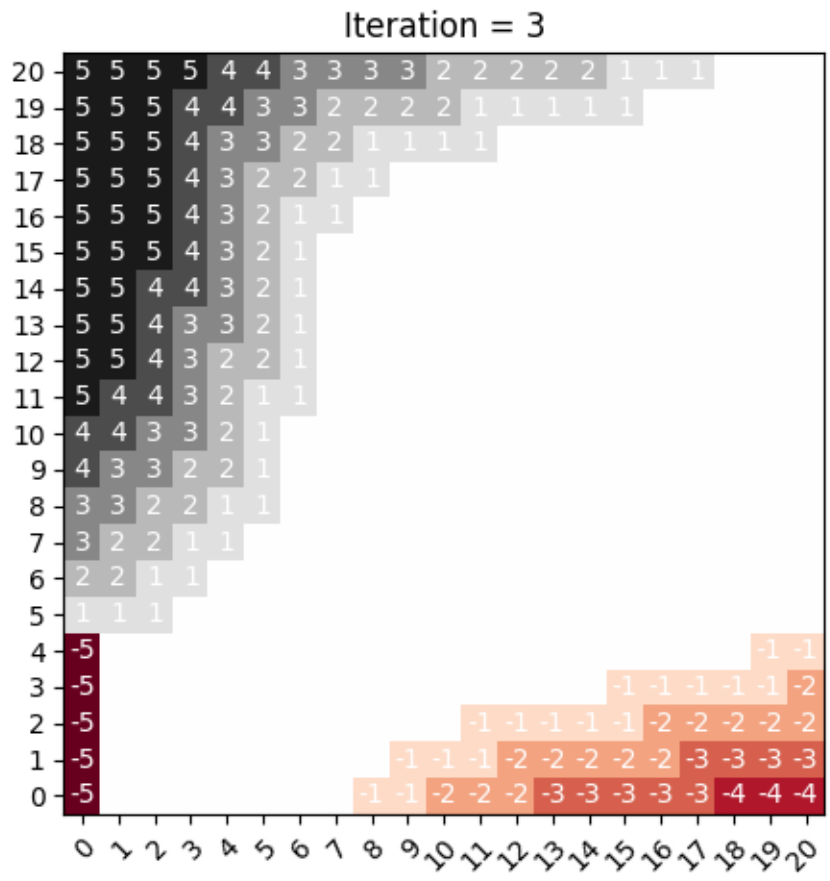
a)

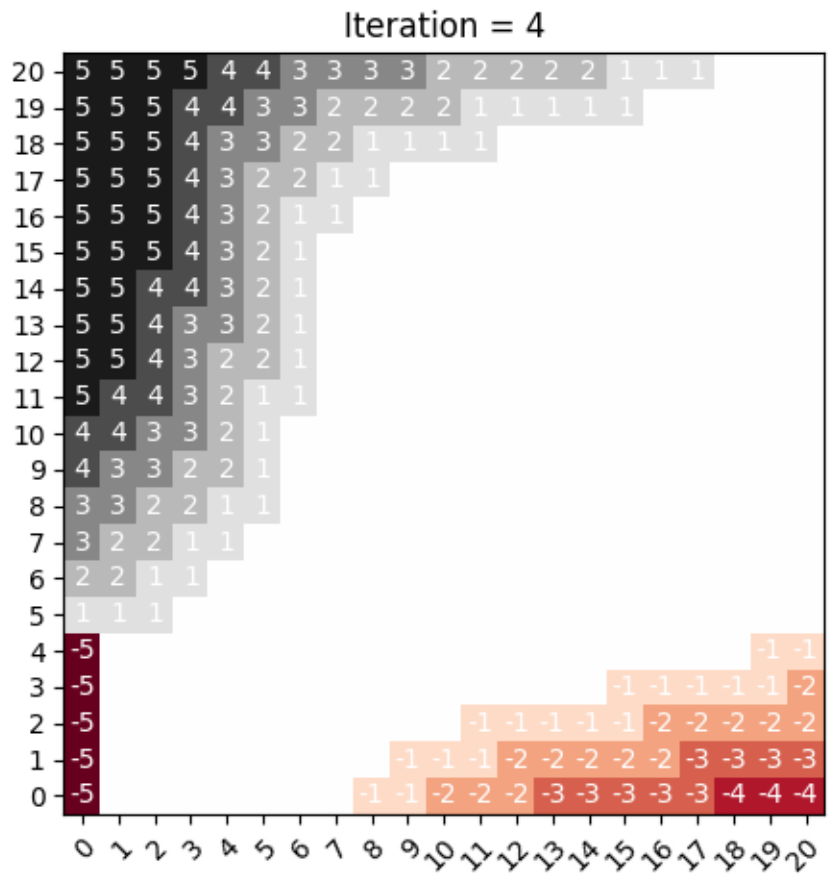



Iteration = 0
20 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 3 2 1
19 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 3 2 1
18 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 3 2 1
17 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 3 3 2 1
16 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 3 2 2 1
15 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 3 3 2 1 1
14 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 3 2 2 1
13 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 3 3 2 1 1
12 5 5 5 5 5 5 5 5 5 5 5 4 4 4 3 3 2 2 1
11 5 5 5 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1 1
10 5 5 5 5 5 5 4 4 4 3 3 3 3 2 2 2 1 1
9 5 5 5 4 4 4 4 3 3 3 2 2 2 2 1 1 1
8 4 4 4 4 3 3 3 3 2 2 2 1 1 1 1
7 4 3 3 3 3 2 2 2 2 1 1 1
6 3 3 2 2 2 2 1 1 1 1
5 2 2 2 1 1 1 1
4 1 1 1 1 -1
3 -5 -1 -1 -1 -1
2 -5 -1 -1 -1 -1 -1 -2 -2 -2
1 -5 -1 -1 -1 -1 -2 -2 -2 -2 -2 -3 -3
0 -5 -1 -1 -1 -2 -2 -2 -2 -3 -3 -3 -3 -3 -4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Iteration = 1
20 5 5 5 4 3 3 2 2 2 1 1 1 1 1 1 1
19 5 5 4 4 3 2 2 1 1 1
18 5 5 4 3 3 2 1 1
17 5 5 4 3 2 2 1
16 5 5 4 3 2 1 1
15 5 5 4 3 2 1
14 5 5 4 3 2 1
13 5 5 4 3 2 1
12 5 5 4 3 2 1
11 5 4 4 3 2 1
10 5 4 3 3 2 1
9 4 4 3 2 2 1
8 3 3 3 2 1 1
7 3 2 2 2 1
6 2 2 1 1 1
5 1 1 1
4 1 -1
3 -5 -1 -1 -1 -1 -1
2 -5 -1 -1 -1 -1 -1 -1 -2 -2 -2 -2
1 -5 -1 -1 -1 -2 -2 -2 -2 -2 -2 -3 -3 -3
0 -5 -1 -1 -1 -2 -2 -2 -3 -3 -3 -3 -3 -3 -4 -4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

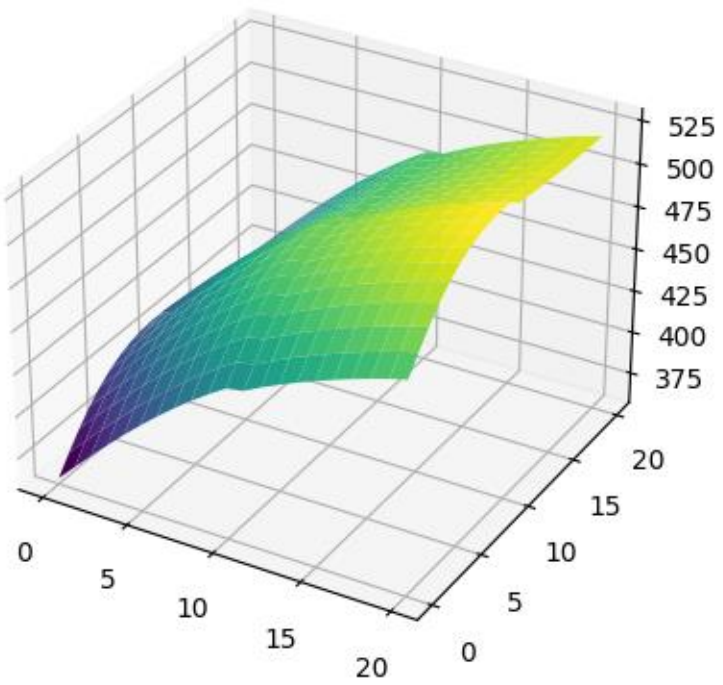

Iteration = 2
20 5 5 5 5 4 4 3 3 3 3 3 2 2 2 2 1 1 1
19 5 5 5 5 4 3 3 2 2 2 2 2 1 1 1 1
18 5 5 5 4 4 3 2 2 1 1 1 1 1
17 5 5 5 4 3 3 2 1 1
16 5 5 5 4 3 2 2 1
15 5 5 5 4 3 2 1 1
14 5 5 4 4 3 2 1
13 5 5 4 3 3 2 1
12 5 5 4 3 2 2 1
11 5 4 4 3 2 1 1
10 4 4 3 3 2 1
9 4 3 3 2 2 1
8 3 3 2 2 1 1
7 3 2 2 1 1
6 2 2 1 1
5 1 1 1
4 -5 -1 -1
3 -5 -1 -1 -1 -1 -1 -2
2 -5 -1 -1 -1 -1 -1 -2 -2 -2 -2 -2
1 -5 -1 -1 -1 -2 -2 -2 -2 -2 -3 -3 -3 -3
0 -5 -1 -1 -2 -2 -2 -3 -3 -3 -3 -3 -4 -4 -4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Iteration = 3
20 5 5 5 5 4 4 3 3 3 3 2 2 2 2 2 1 1 1
19 5 5 5 4 4 3 3 2 2 2 2 1 1 1 1 1
18 5 5 5 4 3 3 2 2 1 1 1 1
17 5 5 5 4 3 2 2 1 1
16 5 5 5 4 3 2 1 1
15 5 5 5 4 3 2 1
14 5 5 4 4 3 2 1
13 5 5 4 3 3 2 1
12 5 5 4 3 2 2 1
11 5 4 4 3 2 1 1
10 4 4 3 3 2 1
9 4 3 3 2 2 1
8 3 3 2 2 1 1
7 3 2 2 1 1
6 2 2 1 1
5 1 1 1
4 -5 -1 -1
3 -5 -1 -1 -1 -1 -1 -2
2 -5 -1 -1 -1 -1 -1 -2 -2 -2 -2 -2
1 -5 -1 -1 -1 -2 -2 -2 -2 -2 -3 -3 -3 -3
0 -5 -1 -1 -2 -2 -2 -3 -3 -3 -3 -3 -4 -4 -4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

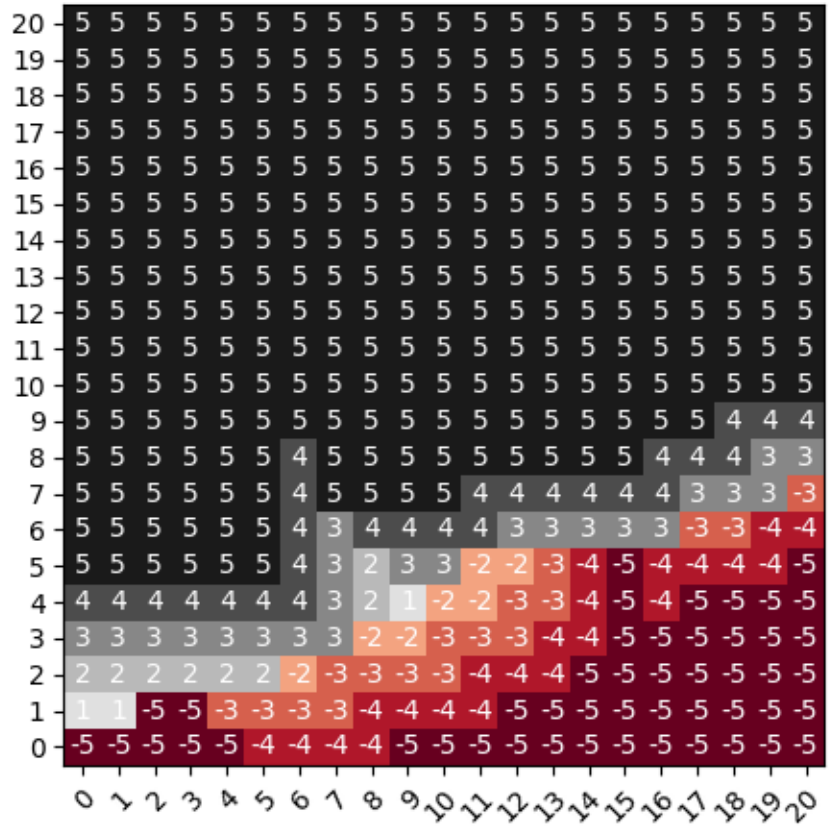

Iteration = 4
20 5 5 5 5 4 4 3 3 3 3 2 2 2 2 2 1 1 1
19 5 5 5 4 4 3 3 2 2 2 2 1 1 1 1 1
18 5 5 5 4 3 3 2 2 1 1 1 1
17 5 5 5 4 3 2 2 1 1
16 5 5 5 4 3 2 1 1
15 5 5 5 4 3 2 1
14 5 5 4 4 3 2 1
13 5 5 4 3 3 2 1
12 5 5 4 3 2 2 1
11 5 4 4 3 2 1 1
10 4 4 3 3 2 1
9 4 3 3 2 2 1
8 3 3 2 2 1 1
7 3 2 2 1 1
6 2 2 1 1
5 1 1 1
4 -5 -1 -1
3 -5 -1 -1 -1 -1 -1 -2
2 -5 -1 -1 -1 -1 -1 -2 -2 -2 -2 -2
1 -5 -1 -1 -1 -2 -2 -2 -2 -2 -3 -3 -3 -3
0 -5 -1 -1 -2 -2 -2 -3 -3 -3 -3 -3 -4 -4 -4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

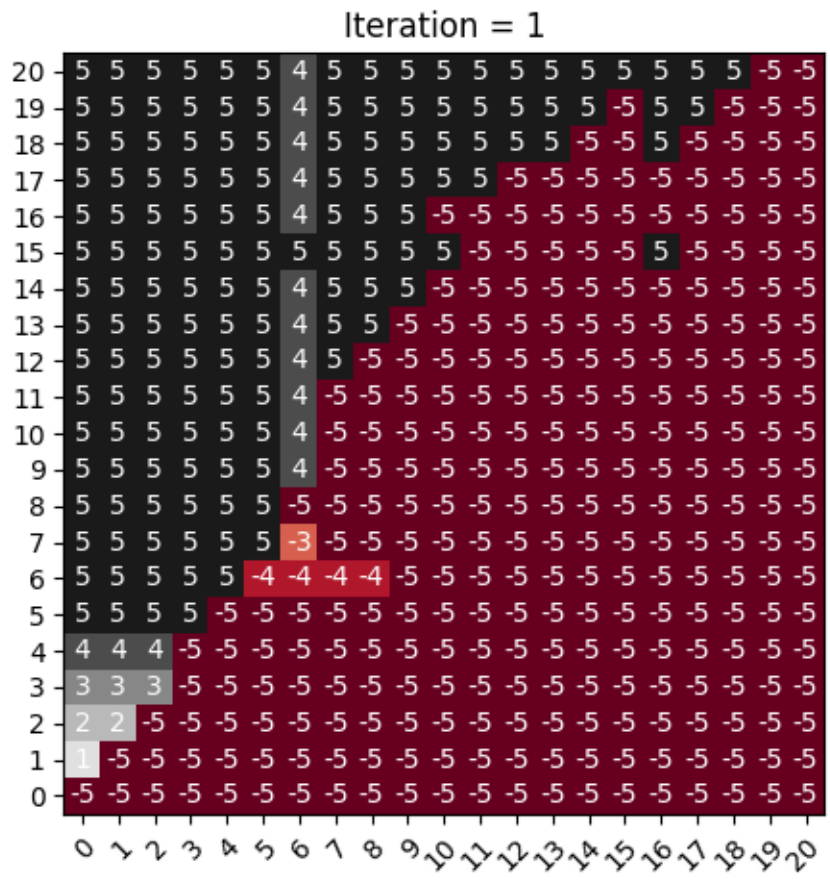
b)

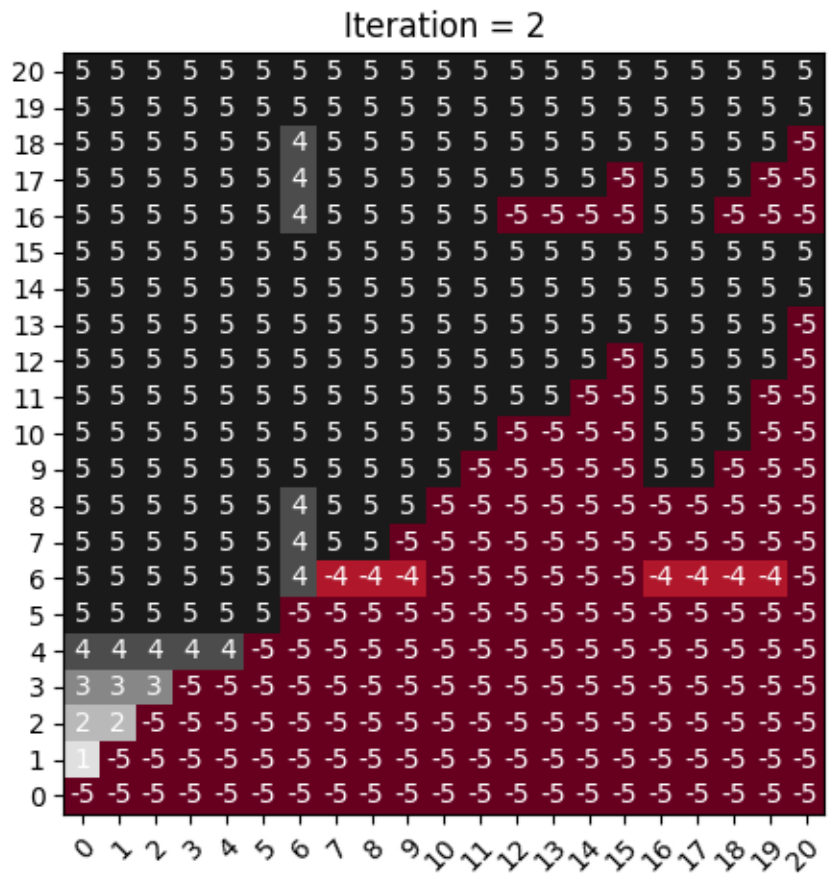
State Value

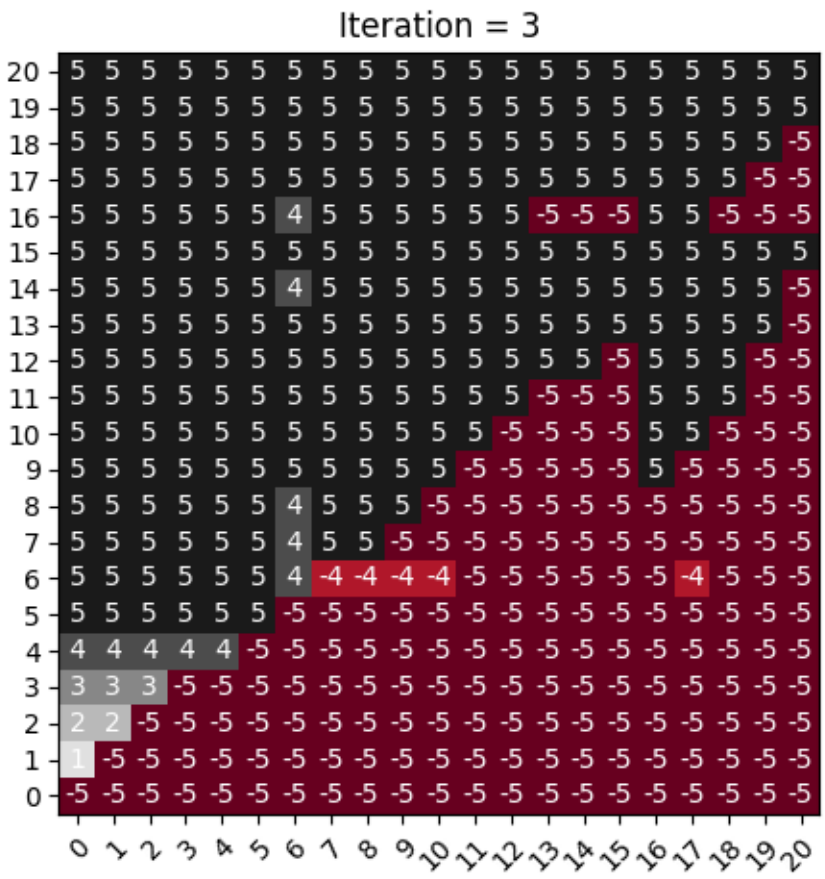


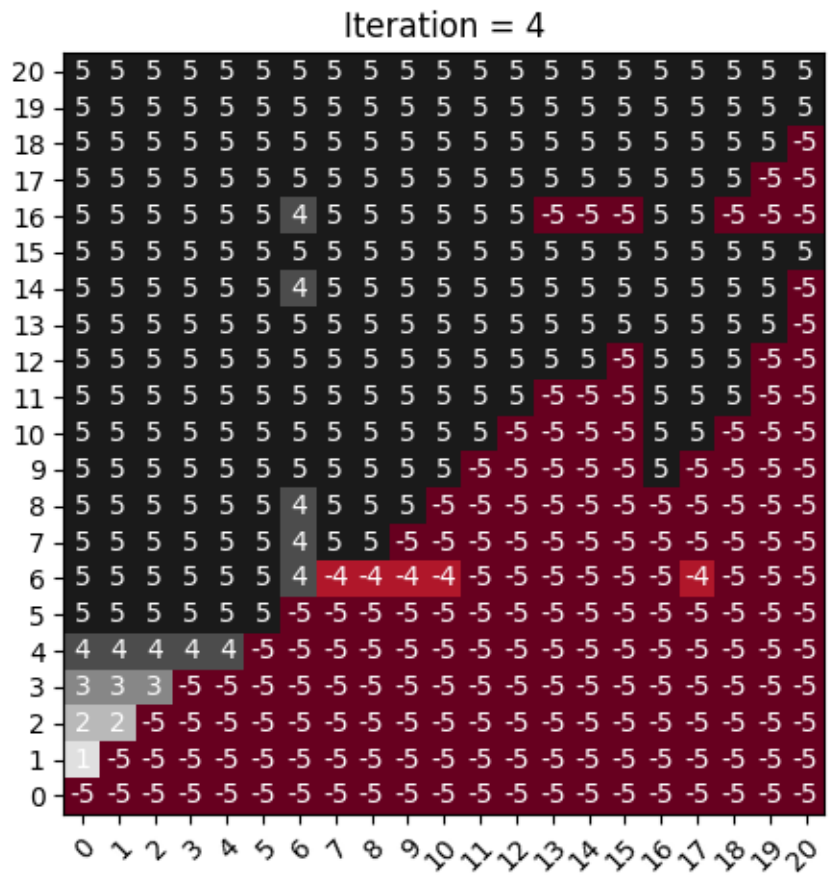
Iteration = 0




Iteration = 1
20 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 5 -5 -5
19 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 -5 5 5 -5 -5 -5
18 5 5 5 5 5 5 4 5 5 5 5 5 5 5 -5 -5 5 -5 -5 -5 -5
17 5 5 5 5 5 5 4 5 5 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5
16 5 5 5 5 5 5 4 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
15 5 5 5 5 5 5 5 5 5 5 5 -5 -5 -5 -5 -5 5 -5 -5 -5 -5
14 5 5 5 5 5 5 4 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
13 5 5 5 5 5 5 4 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
12 5 5 5 5 5 5 4 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
11 5 5 5 5 5 5 4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
10 5 5 5 5 5 5 4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
9 5 5 5 5 5 5 4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
8 5 5 5 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
7 5 5 5 5 5 5 -3 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
6 5 5 5 5 5 -4 -4 -4 -4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
5 5 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
4 4 4 4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
3 3 3 3 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
2 2 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
1 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
0 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Iteration = 2
20 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
19 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
18 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 -5
17 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 -5 5 5 5 -5 -5
16 5 5 5 5 5 5 4 5 5 5 5 5 -5 -5 -5 -5 5 5 -5 -5 -5
15 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
14 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
13 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5
12 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5 5 5 5 5 -5
11 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5 -5 5 5 5 -5 -5
10 5 5 5 5 5 5 5 5 5 5 5 5 5 -5 -5 -5 -5 5 5 5 -5 -5
9 5 5 5 5 5 5 5 5 5 5 5 5 -5 -5 -5 -5 -5 5 5 -5 -5 -5
8 5 5 5 5 5 5 4 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
7 5 5 5 5 5 5 4 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
6 5 5 5 5 5 5 4 -4 -4 -4 -5 -5 -5 -5 -5 -5 -4 -4 -4 -4 -5
5 5 5 5 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
4 4 4 4 4 4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
3 3 3 3 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
2 2 2 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
1 1 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
0 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Iteration = 3
20 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
19 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
18 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5
17 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5 -5
16 5 5 5 5 5 5 4 5 5 5 5 5 5 -5 -5 -5 5 5 -5 -5 -5
15 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
14 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 -5
13 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5
12 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 -5 5 5 5 -5 -5
11 5 5 5 5 5 5 5 5 5 5 5 5 5 -5 -5 -5 5 5 5 -5 -5
10 5 5 5 5 5 5 5 5 5 5 5 5 -5 -5 -5 -5 -5 5 -5 -5 -5
9 5 5 5 5 5 5 5 5 5 5 5 -5 -5 -5 -5 -5 -5 5 -5 -5 -5 -5
8 5 5 5 5 5 5 4 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
7 5 5 5 5 5 5 4 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
6 5 5 5 5 5 5 4 -4 -4 -4 -4 -5 -5 -5 -5 -5 -5 -4 -5 -5 -5
5 5 5 5 5 5 5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
4 4 4 4 4 4 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
3 3 3 3 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
2 2 2 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
1 1 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
0 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



Written:

Preference to Move Cars from First to Second Location: Due to the free transfer of one car from the first location to the second, the optimal policy might show a slight preference towards moving cars in that direction, especially when there are excess cars at the first location.

Avoiding Overflow: Since there's a penalty for having more than 10 cars at a location, the policy will likely prefer actions that avoid causing an overflow, i.e., transferring cars out when the count approaches or exceeds 10.

Balancing Between Costs: The policy needs to strike a balance between the cost of moving cars and the cost of overflow. For instance, if moving several cars avoids the overflow cost, it might be more cost-effective to move the cars, even if it incurs some transfer costs.

The differences in the policy make sense given the new dynamics. The free transfer incentive changes the cost structure and makes certain actions more attractive. The overflow cost acts as a deterrent against keeping too many cars at one location, thus influencing the movement of cars between locations. The policy will now weigh the cost benefits of moving cars versus paying for the overflow, leading to a different optimal strategy.