CS5180 – Ex3

Sudhendra Kambhamettu
NUID: 002786797

Q1.

a. The value function $v_\pi(s)$ under the policy π can be defined in terms of the action-value function as follows:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \cdot q_\pi(s,a)$$

b. The action-value function $q_\pi(s,a)$ can be defined in terms of the state-value function v_π and the transition probabilities pp as:

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

c. The Bellman equation for the action values can be derived as follows:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \,| S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s,a)\,[r + \gamma \mathbb{E}_\pi[G_{t+1}|s', a']] \\ \therefore q_\pi(s,a) &= \sum_{s',r} p(s', r|s,a)\,[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s', a')] \end{aligned}$$

This equation calculates the expected return for taking action a in state s, transitioning to a new state s′, and then continuing according to policy π. The inner sum over a′ accounts for all possible actions a′ that could be taken in the new state s′, weighted by the policy's action probabilities, π(a′|s′), and the expected action values q.

Q2.

a. $v_*(s) = \max_a \big(q_*(s,a)\big)$
b. $q^*(s,a) = \sum_{s',r} p(s', r|s,a)[r + \gamma v^*(s')]$
c. $\pi^*(s) = arg\max_a \big(q_*(s,a)\big)$
d. $\pi^*(s) = arg\max_a \sum_{s',r} p(s', r|s,a)[r + \gamma v_*(s')]$
e. $v_\pi(s) = \sum_a \pi(a|s) \cdot \sum_{a',r} p(s', r|s,a)[r + \gamma v_*(s')]$
    a. $v_*(s') = \max_a \{\sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_*(s')\}$
    b. $q_\pi(s,a) = \sum_{s'} p(s'|s,a)\,[r(s,a) + \gamma \sum_{a'} \pi(a'|s') \cdot q_\pi(s', a')]$
    c. $q_*(s,a) = \sum_{s'} p(s'|s,a)\left[r(s,a) + \gamma \max_{a'} q_*(s', a')\right]$

Q3.

We can change the policy evaluation step to make sure the policy iteration algorithm considers a policy to be stable only if the action it selects for each state maximizes the action-value function qπ(s,a) given the current estimate of the state-value function vπ (s). This will ensure the algorithm always converges. With this adjustment, the algorithm is guaranteed to stop if a policy proves to be optimum after two iterations, as more iterations will not alter the policy.

Adding a check that confirms if the new policy is the same as the policy from the prior iteration after the policy enhancement phase is a workable way to ensure convergence. The algorithm stops if it is. This method necessitates monitoring the policy from the prior iteration and contrasting it with the current iterations policy.

Pseudocode modification:

1. Initialize $v(s)$ arbitrarily for all $s \in S$, and $\pi(s)$ arbitrarily for all $s \in S$
2. Loop for each $s \in S$, update $v(s)$ until the value function vv stabilizes within a threshold θ.
3. Policy stable = true. For each $s \in S$:
    a. $Old\ action\ =\ \pi(s)$.
    b. $\pi^*(s) = arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$
    c. If $old\ action\ \neq \pi(s), then$ policy stable = false.
    d. If after going through all $s \in S$, policy stable remains true, then terminate.

b. Because value iteration changes the value function directly based on the maximizing over action values, it is not affected by the same problem as policy iteration. It is ensured that the value function estimates would improve monotonically with each value iteration, eventually convergent to the ideal value function v^*.

Value iteration does not specifically need a policy to be stable for convergence; instead, it incorporates policy review and improvement into a single phase. Rather, by consistently selecting the course of action that maximizes the expected return according to the current value function estimate, it implicitly determines the best course of action.

Because of this, value iteration does not have an equivalent problem that would lead to policy oscillation or failure to converge under the common assumptions of finite MDPs and discount factors $0 \leq \gamma < 10 \leq \gamma < 1$.

Q4.

a. We modify the normal policy iteration method (which often focuses on state values) to work directly with action values in order to establish policy iteration for action values and compute q^*. The following actions are necessary for this:

    Initialization
    Initialize $q(s, a)$ randomly for all $s \in S\ and\ a \in A$ & initialize a random policy π.

    Policy Evaluation
    Update $q(s, a)$ unitl it converges by using the expected returns following the current policy π. Using the action-value form of the bellman equation:

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

    Policy Improvement
    Make the policy greedier in relation to the existing action-value function q in order to improve it. Select the action (a) that maximizes $q(s, a)$ for every state (s):

    $$\pi^*(s) = arg\max_a \big(q_*(s, a)\big)$$

    Repeat the steps 2 and 3 until the policy does not changes from one iteration to the next i.e., stabilizes at which point $q(s, a) = q^*(s, a)$.

b. The analog of the value iteration update equation for action values, which directly updates **q** values without a separate policy evaluation and improvement phase, can be formulated as follows:

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

This update rule applies value iteration to the action-value space, updating each action value $q_{k+1}(s,a)$ with the maximum of the next state-action pairs' action values, discounted by γ, and added to the expected reward. The procedure repeats until $q_{k+1}(s,a)$ converges to $q^*(s,a)$.

Q5.

Plots

a.

```
[[ 3.30943462  8.789637    4.42795124  5.32266843  1.49249388]
 [ 1.52199138  2.99265921  2.25045007  1.90786176  0.54768952]
 [ 0.0512075   0.73850162  0.67341109  0.35846498 -0.4028696 ]
 [-0.97321735 -0.43517194 -0.3545918  -0.58533367 -1.18281184]
 [-1.85733003 -1.34491138 -1.22898017 -1.42265012 -1.97491941]]
```

b.

```
Iteration:  43
[[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]]

[[[3], [0, 1, 2, 3], [1], [0, 1, 2, 3], [1]], [[0, 3], [0], [0, 1], [1], [1]], [[0, 3], [0], [0, 1], [0, 1], [0, 1]], [[0, 3], [0], [0, 1], [0, 1], [0, 1]], [[0, 3], [0], [0, 1], [0, 1], [0, 1]]]
```

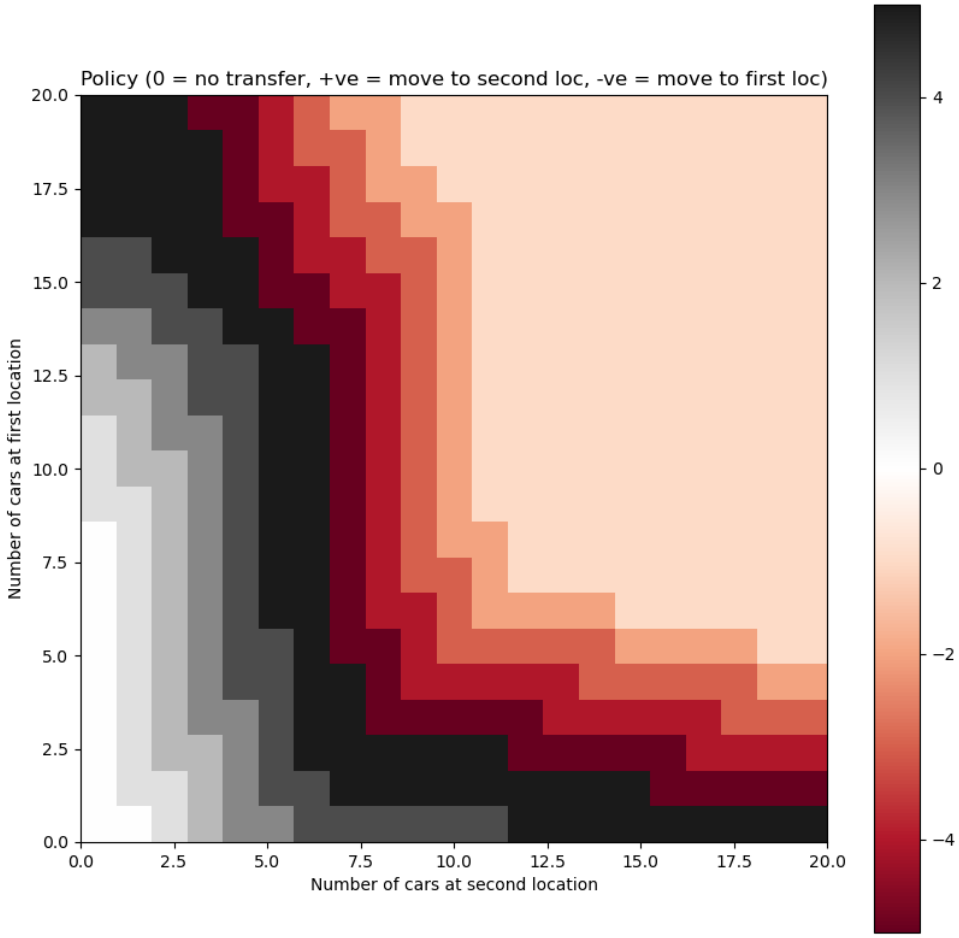
c.

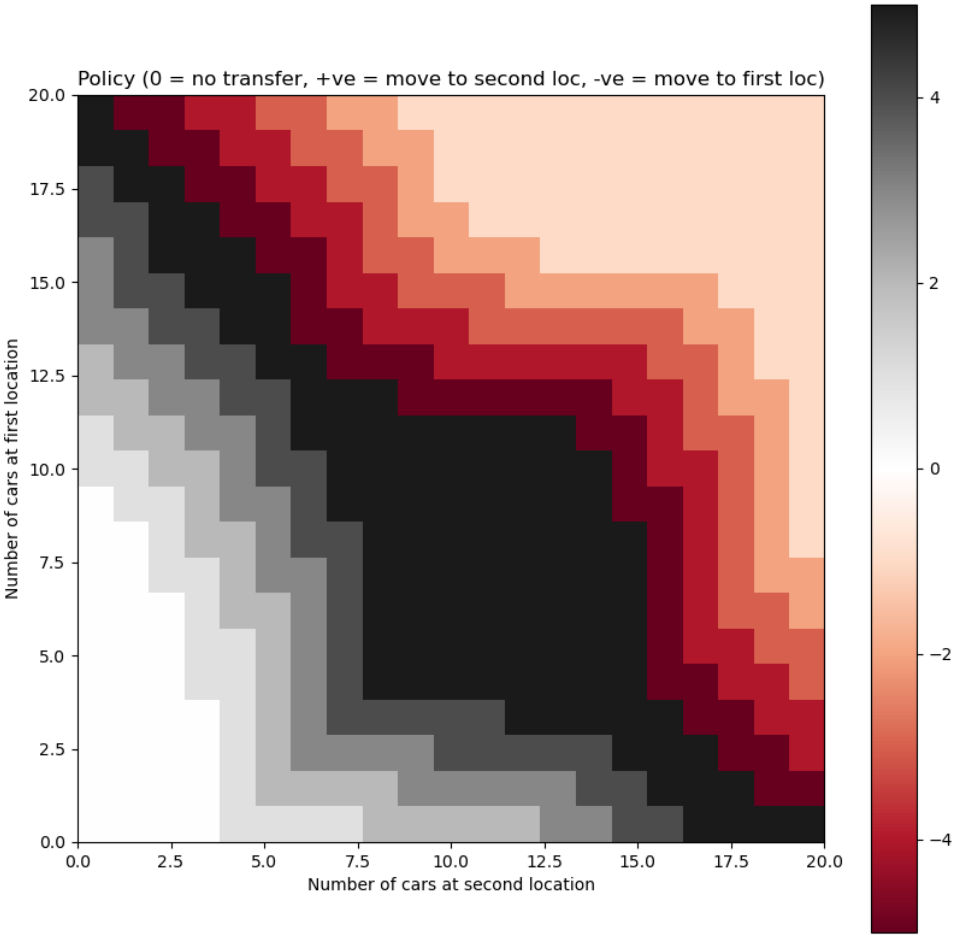
```
[[21.97748528 24.41942809 21.97748528 19.41942809 17.47748528]
 [19.77973675 21.97748528 19.77973675 17.80176308 16.02158677]
 [17.80176307 19.77973675 17.80176308 16.02158677 14.41942809]
 [16.02158677 17.80176308 16.02158677 14.41942809 12.97748528]
 [14.41942809 16.02158677 14.41942809 12.97748528 11.67973676]]
[[[3], [0, 1, 2, 3], [1], [0, 1, 2, 3], [1]], [[3], [0], [0, 1], [1], [1]], [[3], [0], [0, 1], [0, 1], [0, 1]], [[3], [0], [0, 1], [0, 1], [0, 1]], [[3], [0], [0, 1], [0, 1], [0, 1]]]
(explore) PS C:\Users\sudhe\University\CS5180\Assignments\Ex3\ex3-starter> []
```

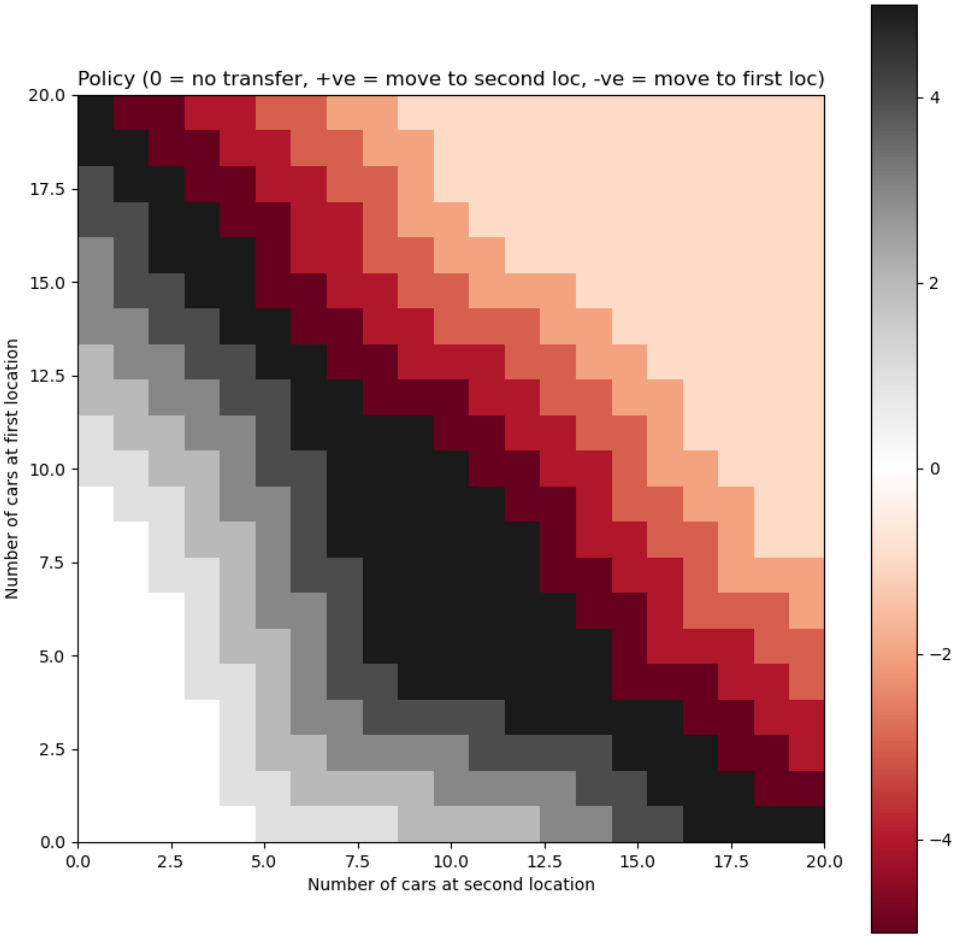
Q6.

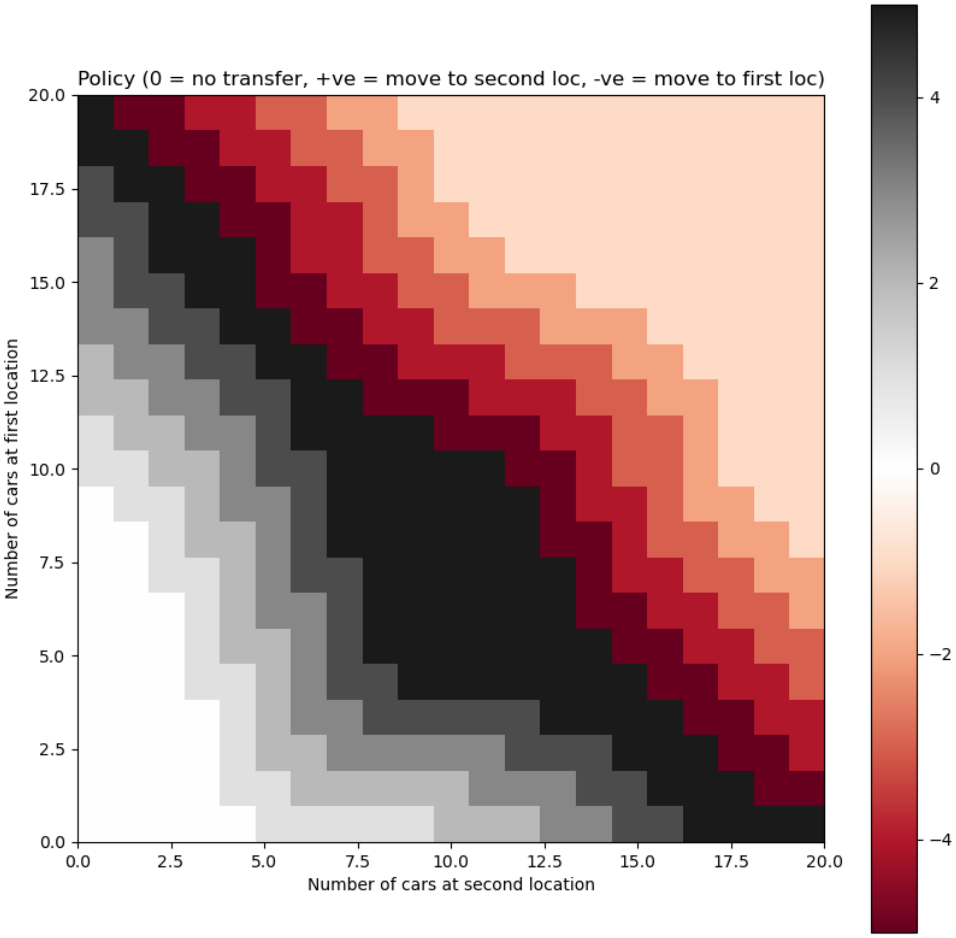
Part a

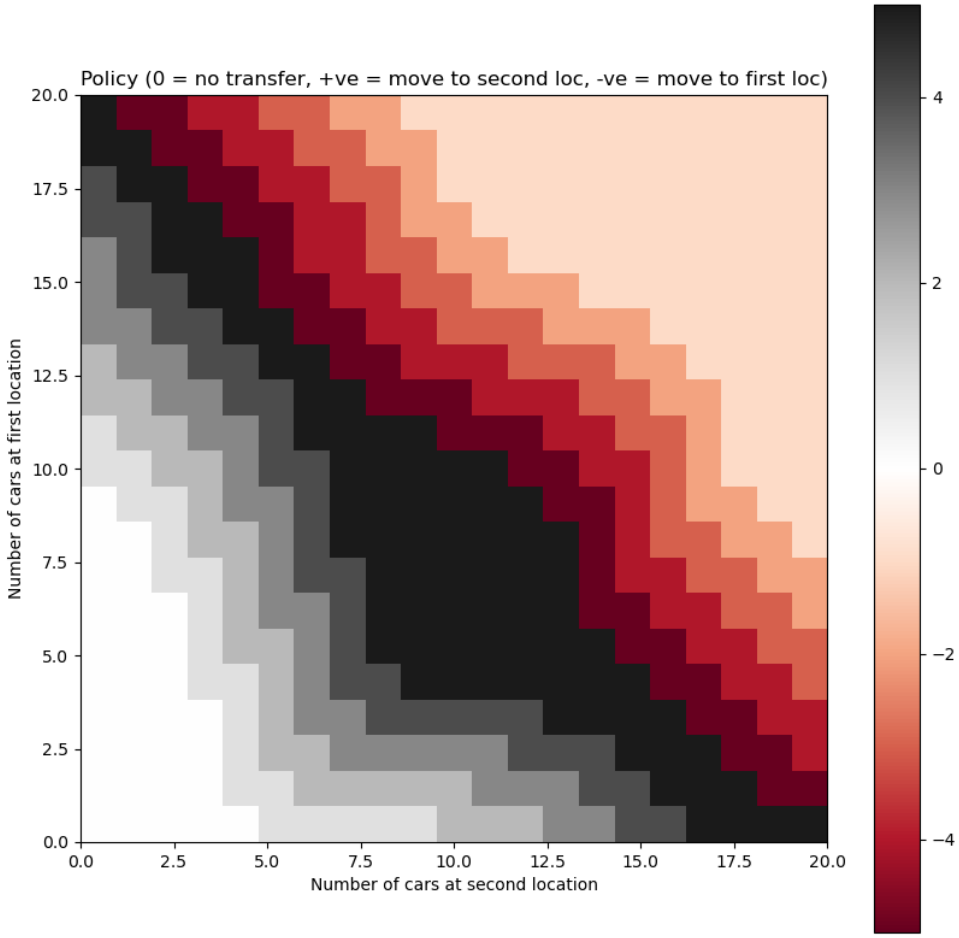
Policy Iteration

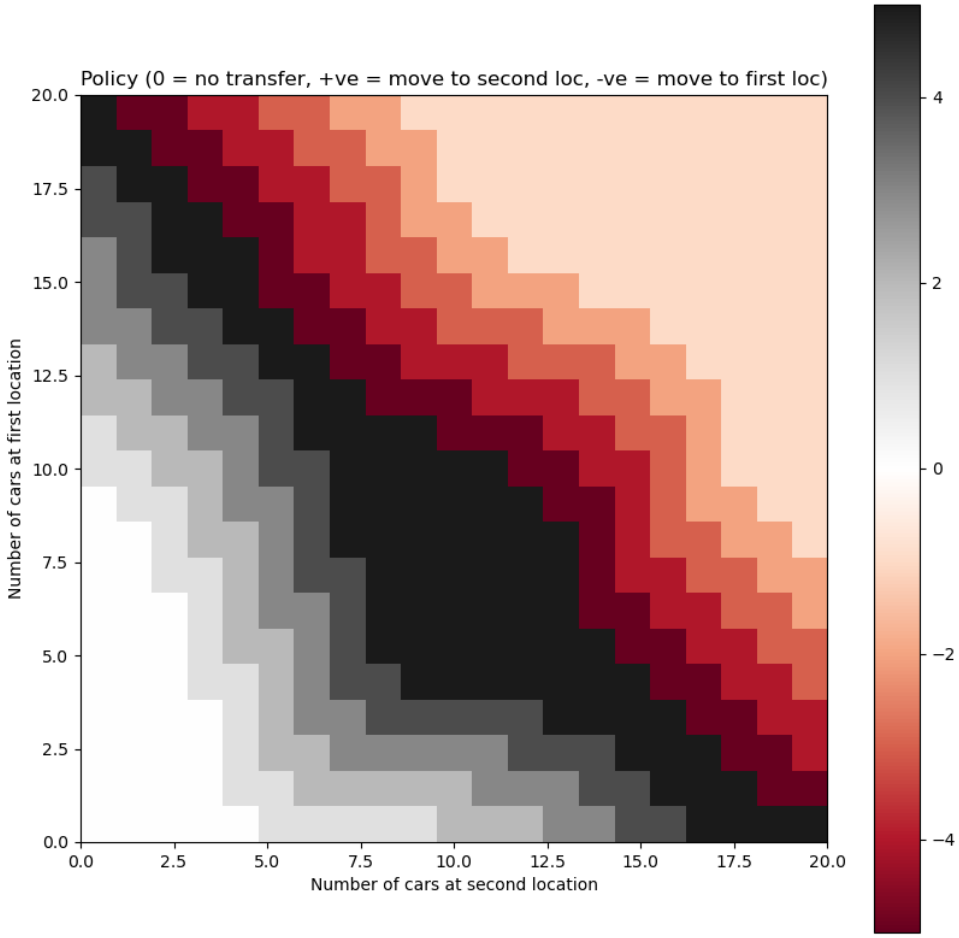

Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4

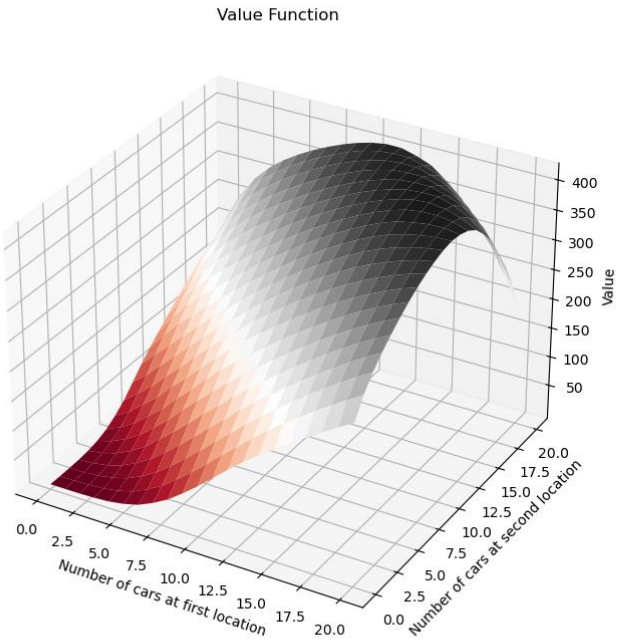

Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4


Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4

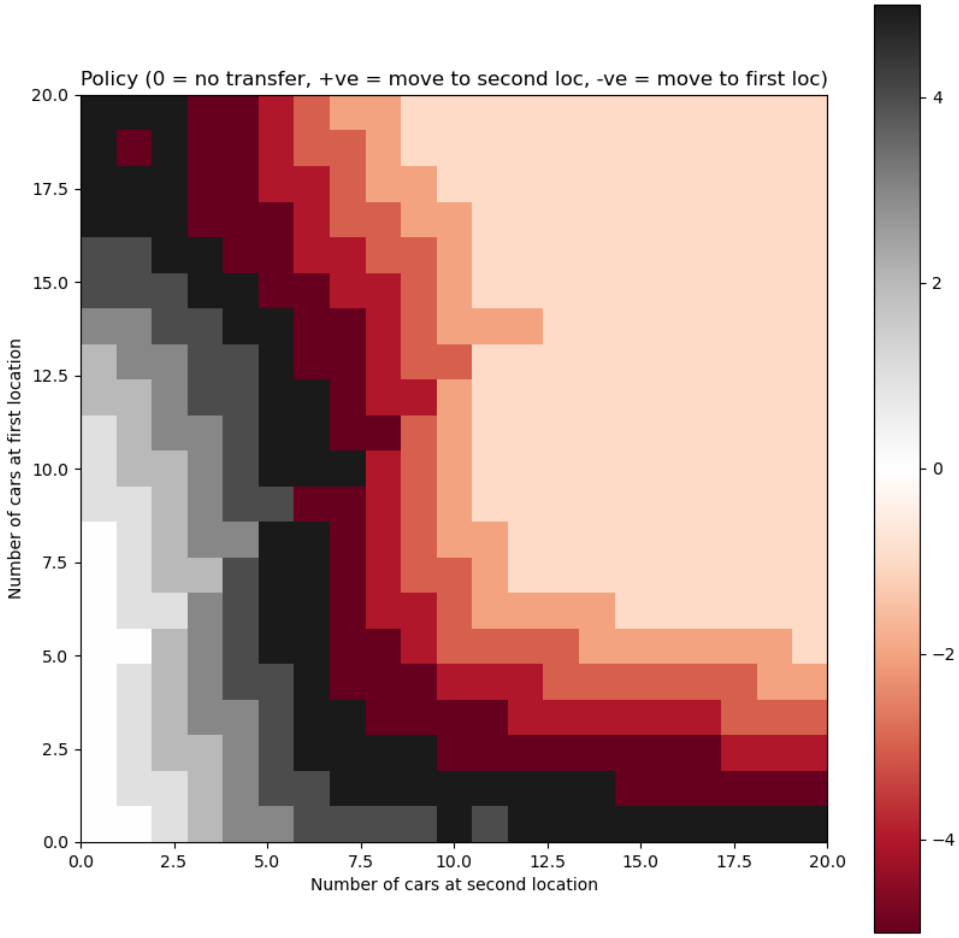

Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4

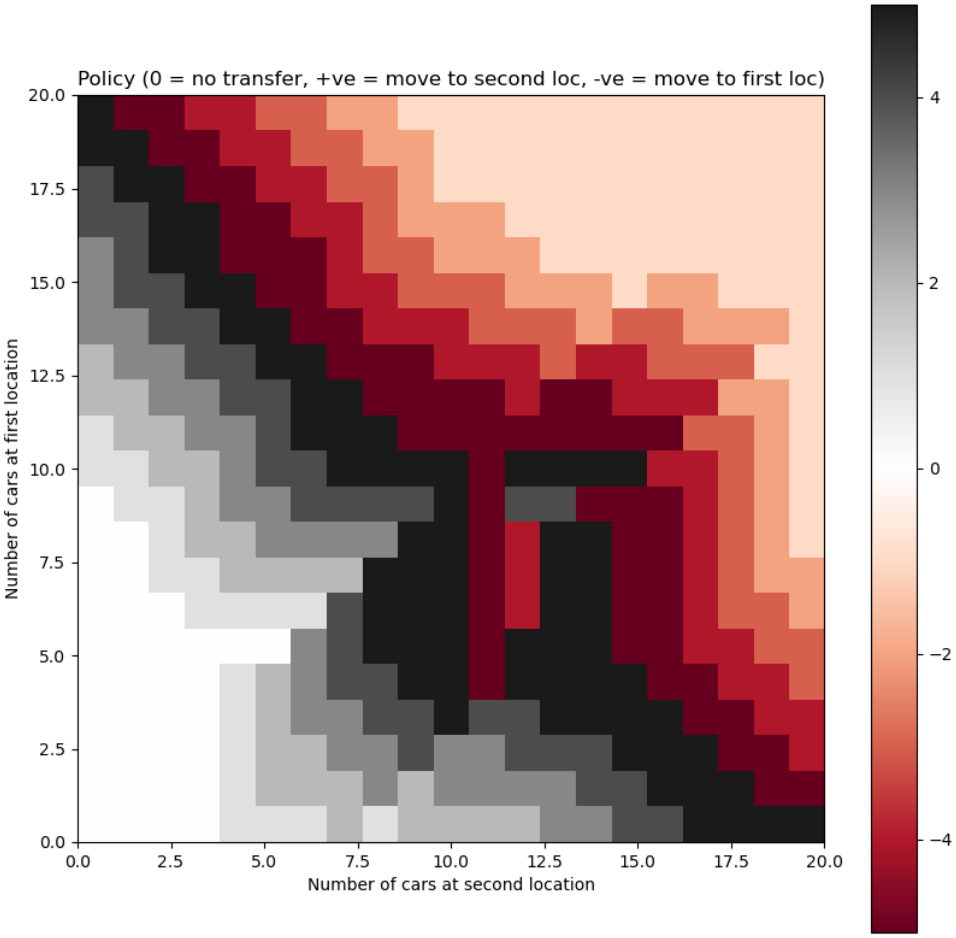

Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4

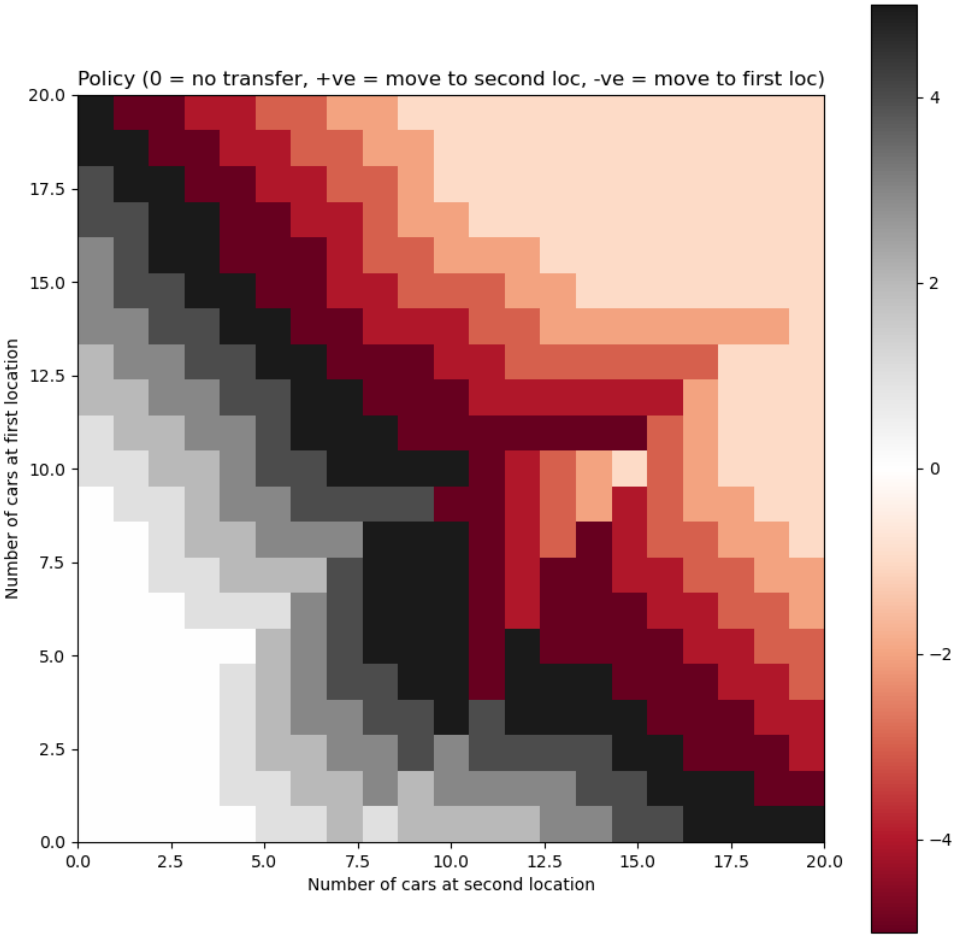

Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4

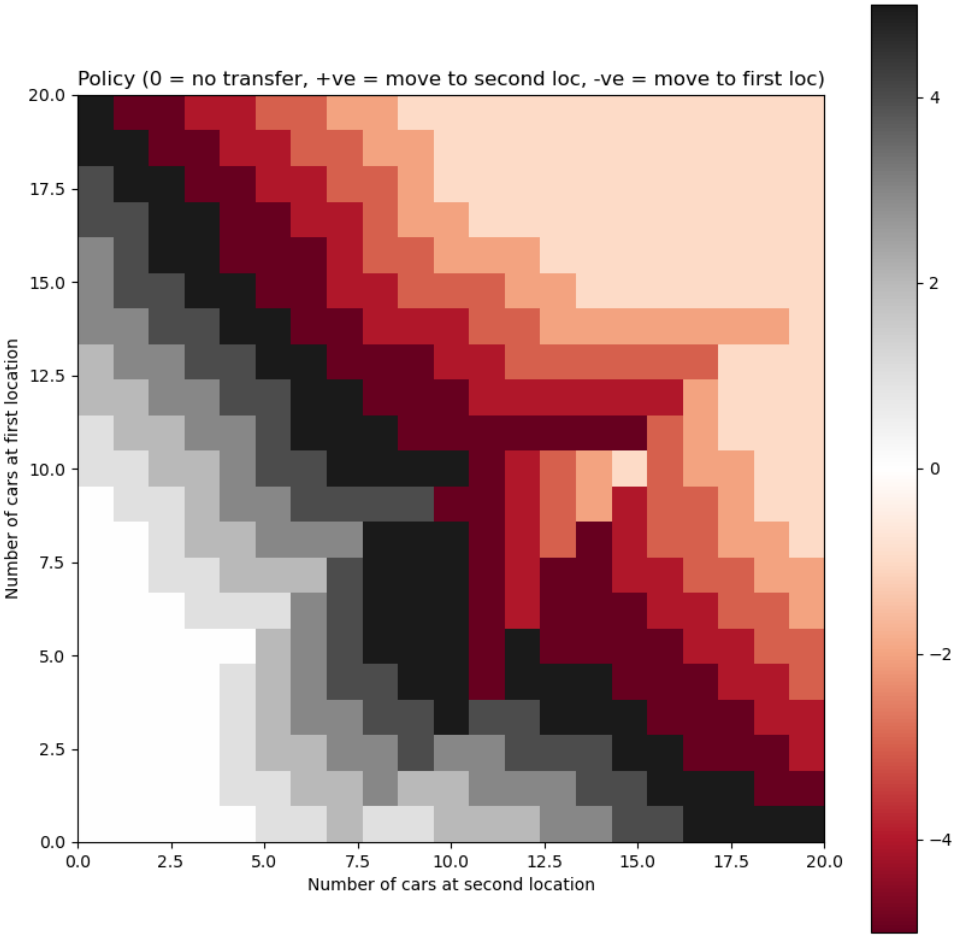

Value Function
400
350
300
250
200
150
100
50
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of cars at first location
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Number of cars at second location

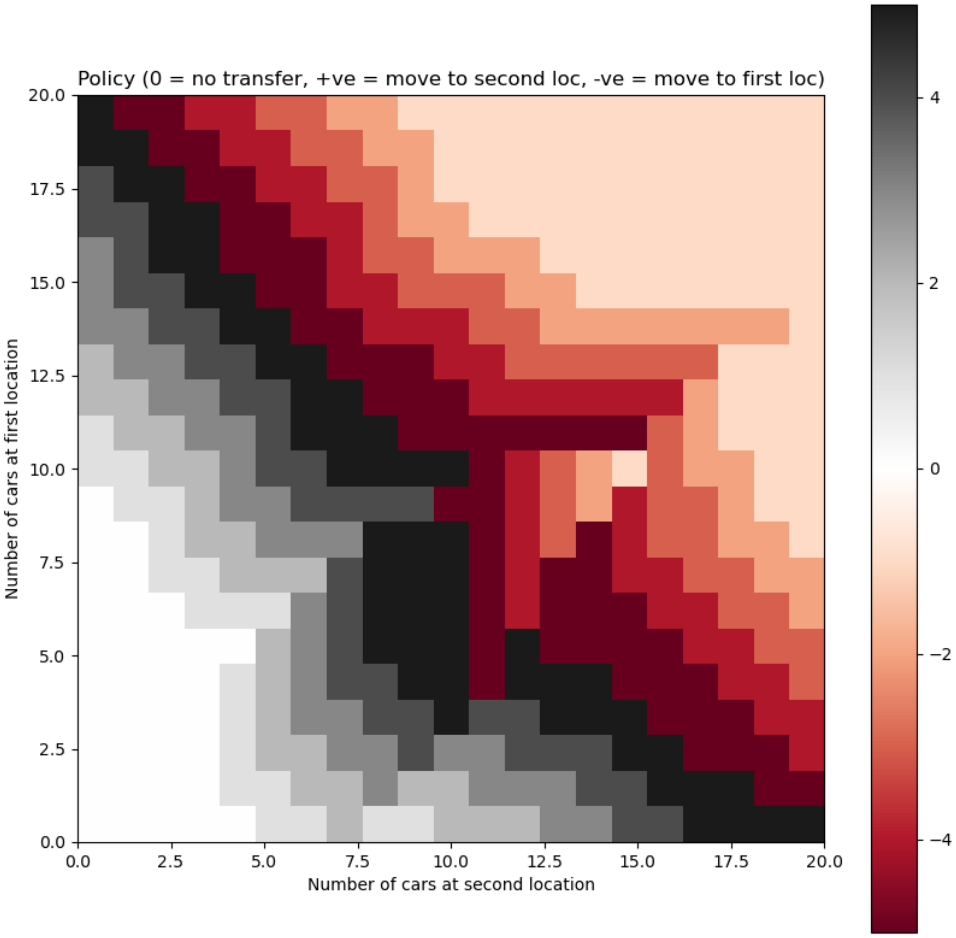
Part b

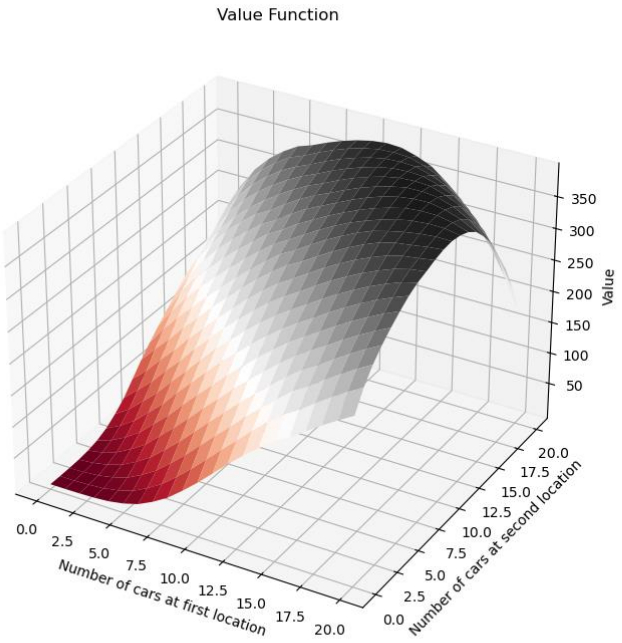

Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4


Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4


Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4


Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4


Policy (0 = no transfer, +ve = move to second loc, -ve = move to first loc)
Number of cars at first location
Number of cars at second location
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
4
2
0
−2
−4



Written 1:

In Q6 part b, the introduction of a free shuttle for moving a car and additional parking costs for more than ten cars at each location significantly alters strategic decisions. The free shuttle incentivizes moving a car from location 1 to location 2 to save costs, especially beneficial when location 2 has fewer than ten cars to avoid extra parking fees. Moreover, the policy adapts to avoid keeping more than ten cars at any location, demonstrating a nuanced strategy to minimize costs and maximize profits by optimizing car distribution across locations.

Written 2:

The modifications in Q6 part b encourage strategic car management to leverage the free shuttle service and mitigate additional parking costs. Specifically, the optimal policy likely shows a preference for redistributing cars to ensure neither location exceeds ten cars, unless the expected rental demand justifies the risk of additional costs. This results in a dynamic strategy where cars are moved not only based on expected demand but also on minimizing operational costs, showcasing the complex interplay between service availability, cost management, and customer demand in the car rental business.

Q7.

a. We need to show that $\max_a f(a) - \max_a g(a) \leq \max_a |f(a) - g(a)|$

Lets consider the two cases based on the sign of the RHS

Case a:

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

Let a_f be the action that maximizes f(a) and a_g the action which maximizes g(a) therefore,

$$f(a_f) \geq g(a_f) \; and \; f(a_g) \geq g(a_g)$$

$$Hence, \max_a f(a) - \max_a g(a) = f(a_f) - g(a_g) \leq f(a_f) - g(a_f) \leq \max_a |f(a) - g(a)|$$

Case b:

$$\max_a f(a) - \max_a g(a) < 0$$

Let a_f and a_g be defined similarly as above. In this case, lets analyze the absolute value, which leads to a similar conclusion that –

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

Combining both the cases we see that regardless of the sign,

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

b. We need to show the following –

$$\left||BV_i - BV_i'|\right|_\infty \leq \gamma ||V_i - V_i'||_\infty$$

The Bellman backup operator applied to the value function V for all states can be written as –

$$BV(s) = \max_a \sum_{s",r} p(s', r|s, a)\,[r + \gamma V(s')]$$

Using the result from part a we can extrapolate the results to apply to any two value function vectors $V_i \; and \; V_i'$ we have:

$$|BV(s) - BV'(s)| \leq \gamma \max_{s'} |V(s') - V'(s')|$$

And taking the maximum over all states gives us the inequality we need –

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

c. Given the result from (b), showing that B is a contraction mapping with contraction factor γ<1, we can prove convergence by the existence of, uniqueness and convergence to a fixed point.

Existence:

The Bellman backup operator B has at least one fixed point x, such that $Bx = x$, due to the completeness of the space of value functions under the $\|\cdot\|_\infty$ norm.

Uniqueness:

Since B is a contraction mapping, its fixed point x is unique.

Convergence:

Value iteration, starting from any initial value function, converges to the fixed point x, which is v^*, the optimal value function as $k \to \infty$.

Here, the proof hinges on the contraction property of the Bellman backup operator and the assumption that $\gamma < 1$, ensuring the value iteration converges to the optimal value function v^*.