Solution:

**4.1**



**4.2**

For the case that the transitions from the original states are unchanged.



So



For the case that the dynamics of state 13 are also changed, such that action down from state 13 takes the agent to the new state 15.



So



**4.3**





**4.4**

We can add a check in step2 to avoid the case that the policy continually switches between two or more policies that are equally good. we can compare the old value with the new policy value, if there are equal, which means the two polices are the same, so we can stop and return the value and policy.

**4.5**

1. Initialization

and arbitrarily for all 

1. Policy evaluation

Loop:



Loop for each :







Until (a small positive number determining the accuracy of estimation)

1. Policy Improvement

policy-stable<-true

For each :

old-action 🡨

🡨

If old-action !=  then policy-stable = false

If policy-stable, thenthen stop and return and ; else go to 2

**4.6**

Because this is a deterministic case, so we need to calculate the expectation instead of deterministic state transition.

Step2’s update should be:

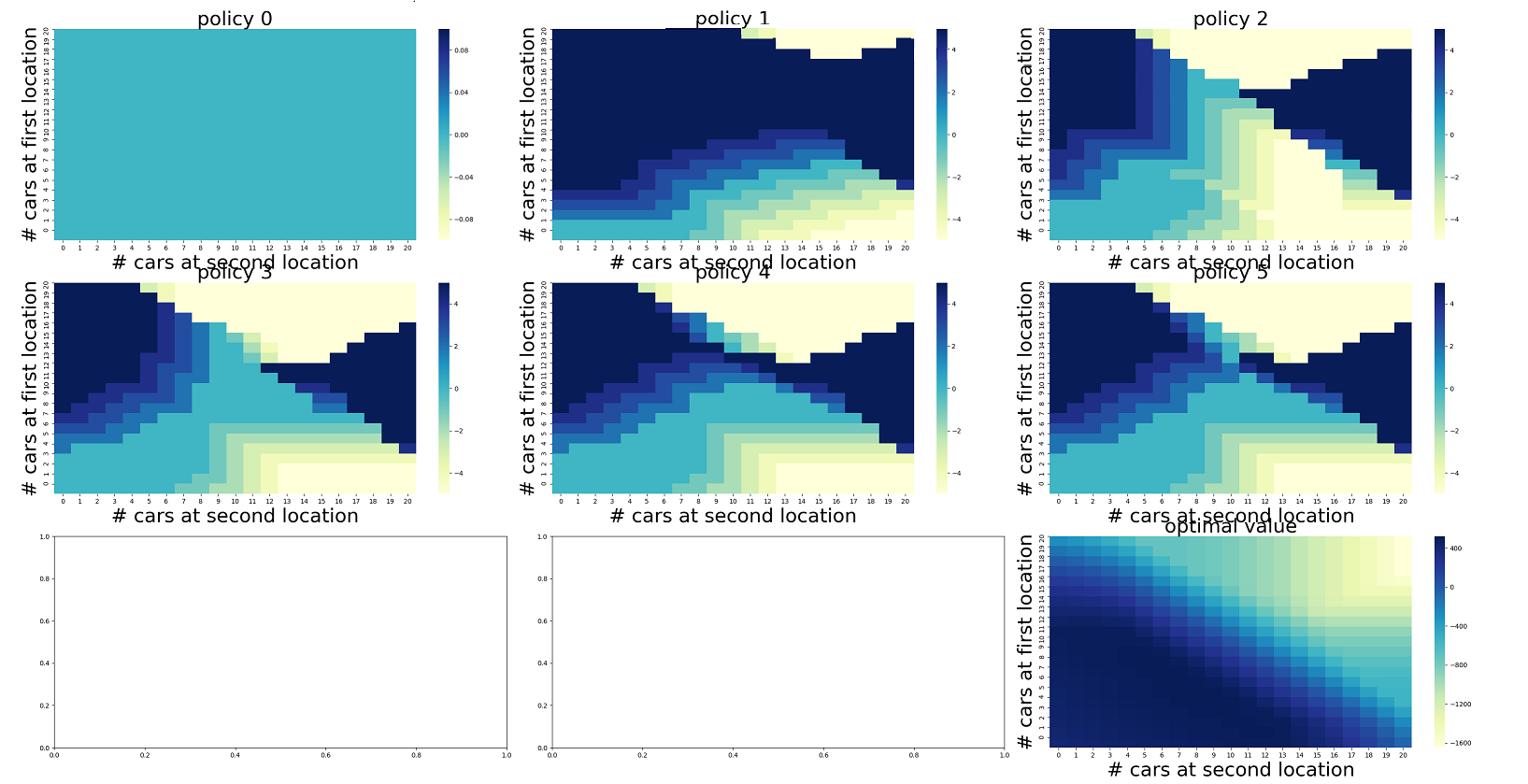


Where



**4.7**

After 6 iteration, it converges to the optimal policy.



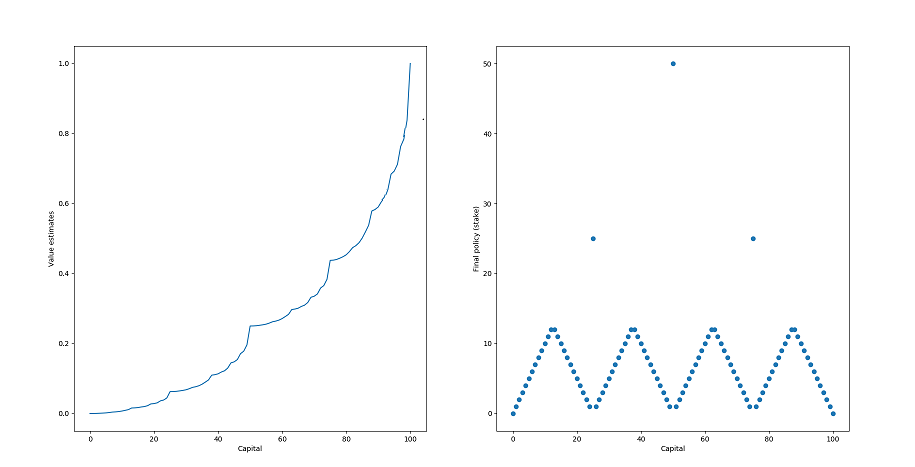
**4.8**

From my perspective, it seem it’s the results of how we calculate the value estimate, which lead to this results. Exactly, A(51) have two optimal action (1, 49).

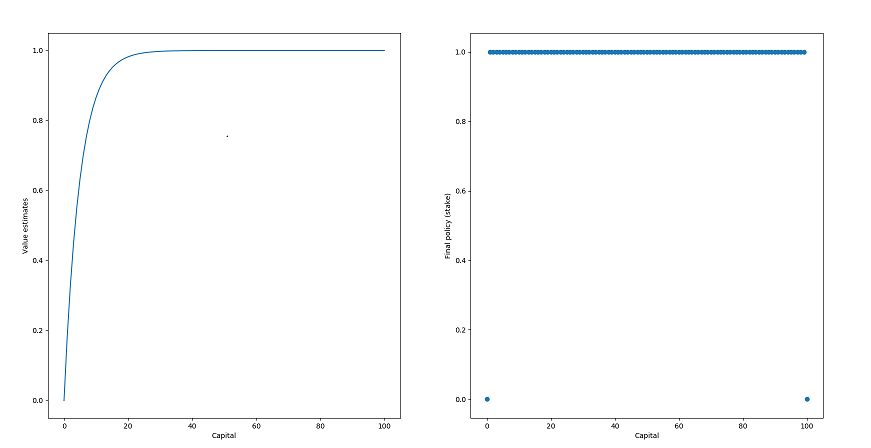
4.9

In the code, the update equation always ignore the Reward, but the state(100) has reward=1 as assumption, why not consider this?

P = 0.25



P = 0.55



4.10

