**Reinforcement Learning**

|  |
| --- |
| **Homework1: Jack’s Car Rental Problem** |
|  |
| **Student:张佳伟**  **ID: 2112103400** |
|  |

# Homework1: Jack’s Car Rental Problem

# Problem description

Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited $10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of $2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number is n is , where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night. We take the discount rate to be γ = 0.9 and formulate this as a continuing finite MDP, where the time steps are days, the state is the number of cars at each location at the end of the day, and the actions are the net numbers of cars moved between the two locations overnight.

# Problem Analysis

we can use the MDP model to solve this problem，We can set up the following parameters：

**State**: the car numer of location 1 and 2, each location is up to 20 cars for rent，so the number of states in this problem is 212

**Action**: Move up to 5 cars from one location to another overnight, so there are 11 actions for this problem.

**Reward**: Jack can make a profit of $10 per car but it must be possible to rent a car

**Transition probability**: The number of cars rented out and the number of cars returned are random, but obey the Poisson distribution.

Renting, returning and moving cars will all have an impact on the final revenue, but in this problem, we can only control and deploy the movement of moving a car. The number of rental and return cars cannot be controlled, so we need to calculate their expectations for optimization iteration.

Policy Iteration and Value Iteration are used to solve the problem, and some approximate simplifications are used to reduce the amount of calculation.

## Problem Resolution

The model of this problem is known and is MDP probelm, so we can solve thsi problem with DP. the Bellman equation is bellow:

## 

when the π(a|s) is confirmed, and beacuse the reward of action is affected by the probability of renting the next day, We can rewrite the formula of state-value update as follows：

We can use the algorithm1 to update state value function with a confirmed π(a|s) , or we can caculate the action return and impove our policy by acting greedily with respect to Vп.The kernel of value function calculation is list below:

|  |
| --- |
| Algorithm1: update-state-value |
| Inputs: 1) state #the car numbers of location A B   1. action #the the moved car overnight 2. state-value   output: the new state-value function   1. initialize: Set v = 0 2. Calculate the cost of action   v ⟵ v   1. Calculate the new state-value function   Loop for rentA in location A:  Loop for rentB in location B:        End for  Return v |

## Policy iteration

For each car in location 1 2, calculate the status values according to the policy, set the value of current statae based on the calculated value. Calculates the absolute value with the greatest change in state value

Then, for each car in location 1 2, impove our policy for the corresponding state by acting greedily.

**Algorithm process:**

|  |
| --- |
| Algorithm2: Policy iteration |
| Inputs: none  output: the optimized policy and best state-value function   1. initialize: Set v = 0, = a small threshold   Set = 0   1. policy evaluation   Loop for each s in States-Set:  Lastv⟵v  v⟵update-state-value     1. policy imporvement   Loop for each s in States-Set:  Loop for each a in Actions-Set:      . |

## Value iteration

In value iteration the solution v\*(s) can be found by one-step lookahead

For each car rented, calculate the value for each action(rent income, car moving cost), Set the value of current state to the maximum action value ,calculates the absolute value with the greatest change in state value.Then, find the most rewarding action and update the policy value for the corresponding state

**Algorithm process**

|  |
| --- |
| Algorithm3: Value iteration |
| Inputs: none  output: the optimized policy and best state-value function   1. initialize: Set v = 0, = a small threshold   Set = 0   1. Single value iteration   Loop for each s in States-Set:  Loop for each a in Actions-Set:           1. select policy   Loop for each s in States-Set:  Loop for each a in Actions-Set: |

# Result

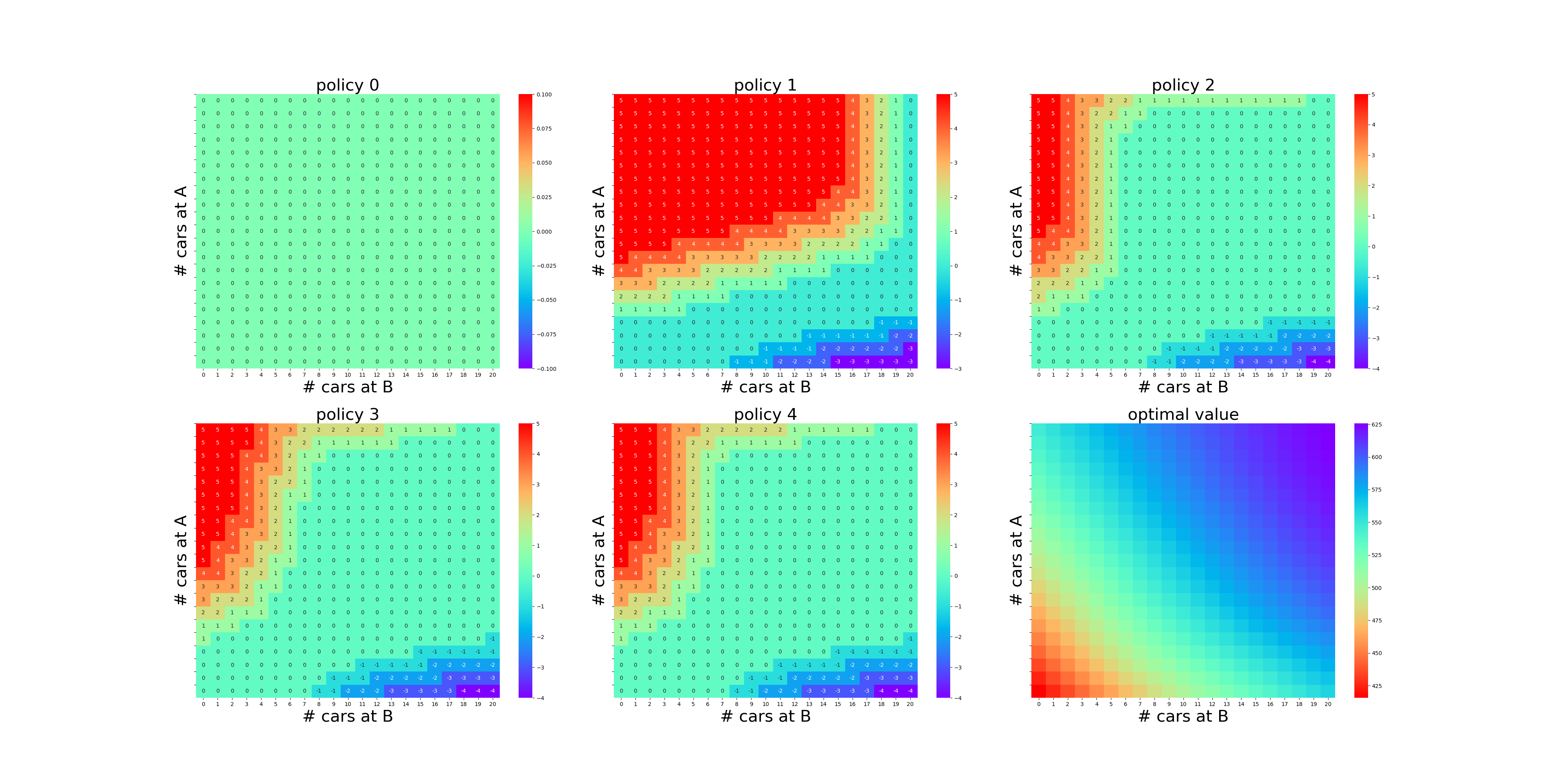


Figure 4.1 The result of policy iteration

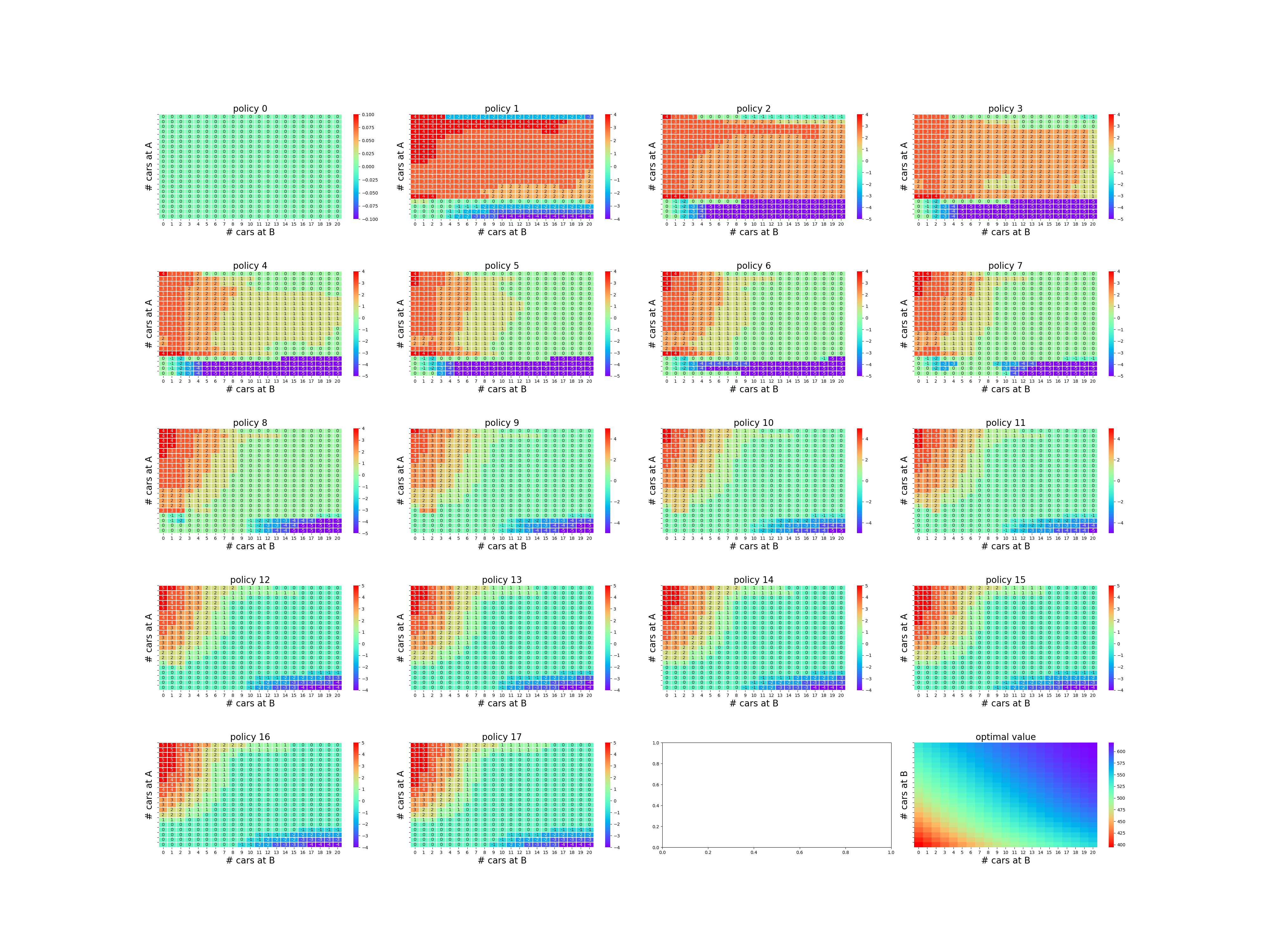


Figure 4.2 The result of value iteration

## Conclusion

Both the value iteration algorithm and the policy iteration algorithm can solve this problem. Compared with the two algorithms, the policy iteration algorithm is faster because it does not need to calculate the reward of all the actions in the value iteration step, but only focuses on one action determined by the policy value.

**Appendix**

