CS 533: Intelligent Agents and Decision Making

Homework #3 Report

Michael Lam
Due: April 20, 2016

This assignment implements an MDP planning algorithm for optimizing expected infinite-horizon discounted cumulative reward and also designs an MDP in the parking domain.

1 Part I

1.1 Design

Two algorithms were implemented: value iteration and policy iteration. Value iteration essentially extends from assignment #2 but returns an approximate value function (albeit bounded). Policy iteration, an algorithm that produces the exact optimal policy and value function, provides a good sanity check for the value iteration algorithm. Exact policy evaluation for policy iteration was also implemented using linear algebra.

1.2 Code

Run the code as follows:

```
python setup.py
source venv/bin/activate
python main.py
deactivate
```

The setup script creates a Python virtual environment and main.py runs the implemented finite hroizon value iteration algorithm on several MDPs. The setup script is not necessary if numpy is already installed on the system.

2 Part II

For the results below, we will provide the value function and policy for the MDP. These will be n-dimensional vectors where indices are the states. To save space, the policy is written as a row vector.

2.1 MDP 1, $\beta = 0.1$

Below is the value function for the MDP with $\beta = 0.1$ using **value iteration** with $\epsilon = 0.000001$. Value iteration converged in 6 iterations. The value function below is within $\epsilon\beta/(1-\beta) = 0.0000001$ factor of the optimal value function.

```
0.10010003
0.00896240
0.00875699
0.00896240
1.00100100
0.00683104
0.06831049
0.00857484
0.01001000
0.08962411
```

Here is the policy for the MDP with $\beta = 0.1$ using **value iteration**:

```
3 3 2 0 0 0 1 2 1 3
```

Here is the value function for the MDP with $\beta = 0.1$ using **policy iteration**:

```
0.10010010
0.00896242
0.00875700
0.00896242
1.00100100
0.00683105
0.06831054
0.00857486
0.01001001
0.08962417
```

Here is the policy for the MDP with $\beta = 0.1$ using **policy iteration**:

```
3 3 2 0 0 0 1 2 1 3
```

2.2 MDP 1, $\beta = 0.9$

Below is the value function for the MDP with $\beta = 0.9$ using **value iteration** with $\epsilon = 0.000001$. Value iteration converged in 131 iterations. The value function below is within $\epsilon\beta/(1-\beta) = 0.000009$ factor of the optimal value function.

```
3.32103018
2.92334785
2.89135152
2.92334785
3.69003357
2.84074234
3.15638060
2.90714616
2.98892716
3.24816429
```

Here is the policy for the MDP with $\beta = 0.9$ using **value iteration**:

```
3 3 2 0 0 0 1 2 1 3
```

Here is the value function for the MDP with $\beta = 0.9$ using **policy iteration**:

```
3.32103321
2.92335061
2.89135433
2.92335061
3.69003690
2.84074523
3.15638358
2.90714894
2.98892989
3.24816735
```

Here is the policy for the MDP with $\beta = 0.9$ using **policy iteration**:

```
3 3 2 0 0 0 1 2 1 3
```

2.3 MDP 2, $\beta = 0.1$

Below is the value function for the MDP with $\beta = 0.1$ using **value iteration** with $\epsilon = 0.000001$. Value iteration converged in 6 iterations. The value function below is within $\epsilon\beta/(1-\beta) = 0.0000001$ factor of the optimal value function.

```
0.01144430
0.01002756
0.57325615
0.00147714
0.06040572
0.10101000
1.01010100
```

```
0.00796657
0.00604054
0.10100563
```

Here is the policy for the MDP with $\beta = 0.1$ using **value iteration**:

```
3 0 0 1 2 2 2 3 1 2
```

Here is the value function for the MDP with $\beta = 0.1$ using **policy iteration**:

```
0.01144435
0.01002757
0.57325622
0.00147723
0.06040579
0.10101010
1.01010101
0.00796659
0.00604058
0.10100573
```

Here is the policy for the MDP with $\beta = 0.1$ using **policy iteration**:

```
3 0 0 1 2 2 2 3 1 2
```

2.4 MDP 2, $\beta = 0.9$

Below is the value function for the MDP with $\beta = 0.9$ using **value iteration** with $\epsilon = 0.000001$. Value iteration converged in 131 iterations. The value function below is within $\epsilon\beta/(1-\beta) = 0.000009$ factor of the optimal value function.

```
4.26315401
4.25764936
5.18851189
3.84402405
4.41685649
4.73683779
5.26315401
3.83510133
3.97517059
4.73683107
```

Here is the policy for the MDP with $\beta = 0.9$ using **value iteration**:

```
0 0 0 1 2 2 2 1 1 2
```

Here is the value function for the MDP with $\beta = 0.9$ using **policy iteration**:

```
4.26315789
4.25765325
5.18851608
3.84402835
4.41686068
4.73684211
5.26315789
3.83510562
3.97517461
4.73683539
```

Here is the policy for the MDP with $\beta = 0.9$ using **policy iteration**:

```
0 0 0 1 2 2 2 1 1 2
```

2.5 Discussion

Since policy iteration with exact policy evaluation yields the exact optimal policy and value function, it can be compared with the approximate value iteration algorithm. We see that the exact optimal value function and approximate value function are within the computed factor, $\epsilon\beta/(1-\beta)$, of each other.

3 Part III

3.1 Design

We designed our MDP as suggested, only we further factorize location L into column $C \in \{A, B\}$ and row $R \in \{1, 2, \ldots, n\}$ where n is the number of parking rows. Therefore our state is a 4-tuple (C, R, O, P), which just makes the location more explicit.

We experimented with two sets of parameters. We made the following assumptions for the first one, which we will call the "reasonable driver":

- The reward for states without being parked is -1 to discourage from driving too much.
- The reward for parking as a function of being in row $r \in \{2, 3, \ldots, n\}$ (note: not in handicap spot) is $(n-(r-1))/(n-1) \cdot F$, where $F = 10$ is an adjustable reward factor. It is more desirable to park nearer to the store.
- The reward for parking in the handicap spots is -100. The assumption here is that the driver really doesn't want to risk a ticket.
- The reward for parking in an occupied spot, i.e. collision, is -10000. Collisions are really, really bad.

- The reward for the terminal state is 1. The terminal state also has no transitions so it's the last state.
- After a DRIVE action, if the next spot is not a handicap spot, the probability that the next spot is occupied by another car is a linear function of the row r: $(n-(r-1))/(n-1)$. As the row goes further away from the store, the probability of being occupied is decreased.
- After a DRIVE action, if the next spot is a handicap spot, the probability that the next spot is occupied by another car is fixed at 0.001, which is a relatively small probability.

The second one, which we will denote as "reckless driver", is the same as above except for the following:

- The reward for parking in a handicap spot is 100. The driver doesn't care about getting a ticket and just wants a spot.
- The reward for parking in an occupied spot, i.e. collision, is 100. The driver is clearly an idiot and doesn't mind causing collisions; he really just wants his parking spot!

Basically the reckless driver just wants a parking spot no matter the consequences. It would make sense that the optimal policies of the reasonable driver and reckless driver would be different. We will test this below.

In all the experiments, we set number of rows $n = 10$. We set the discount factor $\beta = 0.99$, which is close to 1 to let the behavior be mainly influenced by the MDP dynamics and reward. We use policy iteration to get the exact optimal policy and value function.

3.2 Results: reasonable driver

Here is the value function for the reasonable driver:

```
(A, 0, unoccupied, unparked) 85.28853082
(A, 0, unoccupied, parked) -99.01000000
(A, 0, occupied, unparked) 85.28853082
(A, 0, occupied, parked) -9999.01000000
(A, 1, unoccupied, unparked) 89.08010000
(A, 1, unoccupied, parked) 90.99000000
(A, 1, occupied, unparked) 83.43564551
(A, 1, occupied, parked) -9999.01000000
(A, 2, unoccupied, unparked) 82.71889105
(A, 2, unoccupied, parked) 80.99000000
(A, 2, occupied, unparked) 82.71889105
(A, 2, occupied, parked) -9999.01000000
```

```
(A, 3, unoccupied, unparked) 80.89170214
(A, 3, unoccupied, parked) 70.99000000
(A, 3, occupied, unparked) 80.89170214
(A, 3, occupied, parked) -9999.01000000
(A, 4, unoccupied, unparked) 79.08278512
(A, 4, unoccupied, parked) 60.99000000
(A, 4, occupied, unparked) 79.08278512
(A, 4, occupied, parked) -9999.01000000
(A, 5, unoccupied, unparked) 77.29195726
(A, 5, unoccupied, parked) 50.99000000
(A, 5, occupied, unparked) 77.29195726
(A, 5, occupied, parked) -9999.01000000
(A, 6, unoccupied, unparked) 75.51903769
(A, 6, unoccupied, parked) 40.99000000
(A, 6, occupied, unparked) 75.51903769
(A, 6, occupied, parked) -9999.01000000
(A, 7, unoccupied, unparked) 73.76384731
(A, 7, unoccupied, parked) 30.99000000
(A, 7, occupied, unparked) 73.76384731
(A, 7, occupied, parked) -9999.01000000
(A, 8, unoccupied, unparked) 72.02620884
(A, 8, unoccupied, parked) 20.99000000
(A, 8, occupied, unparked) 72.02620884
(A, 8, occupied, parked) -9999.01000000
(A, 9, unoccupied, unparked) 70.30594675
(A, 9, unoccupied, parked) 10.99000000
(A, 9, occupied, unparked) 70.30594675
(A, 9, occupied, parked) -9999.01000000
(B, 0, unoccupied, unparked) 87.16013214
(B, 0, unoccupied, parked) -99.01000000
(B, 0, occupied, unparked) 87.16013214
(B, 0, occupied, parked) -9999.01000000
(B, 1, unoccupied, unparked) 89.08010000
(B, 1, unoccupied, parked) 90.99000000
(B, 1, occupied, unparked) 59.61862877
(B, 1, occupied, parked) -9999.01000000
(B, 2, unoccupied, unparked) 79.18010000
(B, 2, unoccupied, parked) 80.99000000
(B, 2, occupied, unparked) 59.23658684
(B, 2, occupied, parked) -9999.01000000
(B, 3, unoccupied, unparked) 69.28010000
(B, 3, unoccupied, parked) 70.99000000
(B, 3, occupied, unparked) 58.73627151
(B, 3, occupied, parked) -9999.01000000
```

```
(B, 4, unoccupied, unparked) 60.33966819
(B, 4, unoccupied, parked) 60.99000000
(B, 4, occupied, unparked) 60.33966819
(B, 4, occupied, parked) -9999.01000000
(B, 5, unoccupied, unparked) 61.95926080
(B, 5, unoccupied, parked) 50.99000000
(B, 5, occupied, unparked) 61.95926080
(B, 5, occupied, parked) -9999.01000000
(B, 6, unoccupied, unparked) 63.59521293
(B, 6, unoccupied, parked) 40.99000000
(B, 6, occupied, unparked) 63.59521293
(B, 6, occupied, parked) -9999.01000000
(B, 7, unoccupied, unparked) 65.24768983
(B, 7, unoccupied, parked) 30.99000000
(B, 7, occupied, unparked) 65.24768983
(B, 7, occupied, parked) -9999.01000000
(B, 8, unoccupied, unparked) 66.91685841
(B, 8, unoccupied, parked) 20.99000000
(B, 8, occupied, unparked) 66.91685841
(B, 8, occupied, parked) -9999.01000000
(B, 9, unoccupied, unparked) 68.60288729
(B, 9, unoccupied, parked) 10.99000000
(B, 9, occupied, unparked) 68.60288729
(B, 9, occupied, parked) -9999.01000000
(terminal state) 1.00000000
```

The value function has negative values for the collision states and parking in the handicap spots. The collision states are more negative than parking in the handicap spots. The highest value is 90.99, which corresponds to states where the agent has parked in an unoccupied spot in A1 or B1. Thus these are the best spots to park. The values decrease as the row number gets larger, which makes sense since the reward for parking farther from the store is less.

Here is the policy for the reasonable driver:

```
(A, 0, unoccupied, unparked) DRIVE
(A, 0, unoccupied, parked) EXIT
(A, 0, occupied, unparked) DRIVE
(A, 0, occupied, parked) EXIT
(A, 1, unoccupied, unparked) PARK
(A, 1, unoccupied, parked) EXIT
(A, 1, occupied, unparked) DRIVE
(A, 1, occupied, parked) EXIT
(A, 2, unoccupied, unparked) DRIVE
(A, 2, unoccupied, parked) EXIT
```

```
(A, 2, occupied, unparked) DRIVE
(A, 2, occupied, parked) EXIT
(A, 3, unoccupied, unparked) DRIVE
(A, 3, unoccupied, parked) EXIT
(A, 3, occupied, unparked) DRIVE
(A, 3, occupied, parked) EXIT
(A, 4, unoccupied, unparked) DRIVE
(A, 4, unoccupied, parked) EXIT
(A, 4, occupied, unparked) DRIVE
(A, 4, occupied, parked) EXIT
(A, 5, unoccupied, unparked) DRIVE
(A, 5, unoccupied, parked) EXIT
(A, 5, occupied, unparked) DRIVE
(A, 5, occupied, parked) EXIT
(A, 6, unoccupied, unparked) DRIVE
(A, 6, unoccupied, parked) EXIT
(A, 6, occupied, unparked) DRIVE
(A, 6, occupied, parked) EXIT
(A, 7, unoccupied, unparked) DRIVE
(A, 7, unoccupied, parked) EXIT
(A, 7, occupied, unparked) DRIVE
(A, 7, occupied, parked) EXIT
(A, 8, unoccupied, unparked) DRIVE
(A, 8, unoccupied, parked) EXIT
(A, 8, occupied, unparked) DRIVE
(A, 8, occupied, parked) EXIT
(A, 9, unoccupied, unparked) DRIVE
(A, 9, unoccupied, parked) EXIT
(A, 9, occupied, unparked) DRIVE
(A, 9, occupied, parked) EXIT
(B, 0, unoccupied, unparked) DRIVE
(B, 0, unoccupied, parked) EXIT
(B, 0, occupied, unparked) DRIVE
(B, 0, occupied, parked) EXIT
(B, 1, unoccupied, unparked) PARK
(B, 1, unoccupied, parked) EXIT
(B, 1, occupied, unparked) DRIVE
(B, 1, occupied, parked) EXIT
(B, 2, unoccupied, unparked) PARK
(B, 2, unoccupied, parked) EXIT
(B, 2, occupied, unparked) DRIVE
(B, 2, occupied, parked) EXIT
(B, 3, unoccupied, unparked) PARK
(B, 3, unoccupied, parked) EXIT
```

```
(B, 3, occupied, unparked) DRIVE
(B, 3, occupied, parked) EXIT
(B, 4, unoccupied, unparked) DRIVE
(B, 4, unoccupied, parked) EXIT
(B, 4, occupied, unparked) DRIVE
(B, 4, occupied, parked) EXIT
(B, 5, unoccupied, unparked) DRIVE
(B, 5, unoccupied, parked) EXIT
(B, 5, occupied, unparked) DRIVE
(B, 5, occupied, parked) EXIT
(B, 6, unoccupied, unparked) DRIVE
(B, 6, unoccupied, parked) EXIT
(B, 6, occupied, unparked) DRIVE
(B, 6, occupied, parked) EXIT
(B, 7, unoccupied, unparked) DRIVE
(B, 7, unoccupied, parked) EXIT
(B, 7, occupied, unparked) DRIVE
(B, 7, occupied, parked) EXIT
(B, 8, unoccupied, unparked) DRIVE
(B, 8, unoccupied, parked) EXIT
(B, 8, occupied, unparked) DRIVE
(B, 8, occupied, parked) EXIT
(B, 9, unoccupied, unparked) DRIVE
(B, 9, unoccupied, parked) EXIT
(B, 9, occupied, unparked) DRIVE
(B, 9, occupied, parked) EXIT
(terminal state) N/A
```

The policy makes sense. When parked, the next action is EXIT. When not parked, the action is either DRIVE or PARK. Here we see that when the spot is occupied by another car, we see PARK as the optimal action. This makes sense to avoid a collision.

Now what's interesting is the decision to DRIVE or PARK when the parking spot is unoccupied. The optimal action at a handicap spot is to DRIVE. This makes sense since the handicap spot has a negative reward. The spots where the optimal action is to PARK are A1, B1, B2 and B3. This is interesting: why not A2 and A3 as well? Perhaps due to the clockwise circular path of the driving, the agent is betting that it can score better with A1 and B1, and if it can't, it can at least take B2 and B3. That seems to be a plausible explanation.

3.3 Results: reckless driver

Here is the value function for the reckless driver:

```
(A, 0, unoccupied, unparked) 98.98010000
```

```
(A, 0, unoccupied, parked) 100.99000000
(A, 0, occupied, unparked) 98.98010000
(A, 0, occupied, parked) 100.99000000
(A, 1, unoccupied, unparked) 96.99029900
(A, 1, unoccupied, parked) 90.99000000
(A, 1, occupied, unparked) 98.98010000
(A, 1, occupied, parked) 100.99000000
(A, 2, unoccupied, unparked) 96.59631840
(A, 2, unoccupied, parked) 80.99000000
(A, 2, occupied, unparked) 98.98010000
(A, 2, occupied, parked) 100.99000000
(A, 3, unoccupied, unparked) 96.28231587
(A, 3, unoccupied, parked) 70.99000000
(A, 3, occupied, unparked) 98.98010000
(A, 3, occupied, parked) 100.99000000
(A, 4, unoccupied, unparked) 95.92197648
(A, 4, unoccupied, parked) 60.99000000
(A, 4, occupied, unparked) 98.98010000
(A, 4, occupied, parked) 100.99000000
(A, 5, unoccupied, unparked) 95.47652786
(A, 5, unoccupied, parked) 50.99000000
(A, 5, occupied, unparked) 98.98010000
(A, 5, occupied, parked) 100.99000000
(A, 6, unoccupied, unparked) 94.90917715
(A, 6, unoccupied, parked) 40.99000000
(A, 6, occupied, unparked) 98.98010000
(A, 6, occupied, parked) 100.99000000
(A, 7, unoccupied, unparked) 94.16914946
(A, 7, unoccupied, parked) 30.99000000
(A, 7, occupied, unparked) 98.98010000
(A, 7, occupied, parked) 100.99000000
(A, 8, unoccupied, unparked) 93.18002618
(A, 8, unoccupied, parked) 20.99000000
(A, 8, occupied, unparked) 98.98010000
(A, 8, occupied, parked) 100.99000000
(A, 9, unoccupied, unparked) 91.82243322
(A, 9, unoccupied, parked) 10.99000000
(A, 9, occupied, unparked) 98.98010000
(A, 9, occupied, parked) 100.99000000
(B, 0, unoccupied, unparked) 98.98010000
(B, 0, unoccupied, parked) 100.99000000
(B, 0, occupied, unparked) 98.98010000
(B, 0, occupied, parked) 100.99000000
(B, 1, unoccupied, unparked) 96.73046572
```

```
(B, 1, unoccupied, parked) 90.99000000
(B, 1, occupied, unparked) 98.98010000
(B, 1, occupied, parked) 100.99000000
(B, 2, unoccupied, unparked) 96.35552145
(B, 2, unoccupied, parked) 80.99000000
(B, 2, occupied, unparked) 98.98010000
(B, 2, occupied, parked) 100.99000000
(B, 3, unoccupied, unparked) 95.77415280
(B, 3, unoccupied, parked) 70.99000000
(B, 3, occupied, unparked) 98.98010000
(B, 3, occupied, parked) 100.99000000
(B, 4, unoccupied, unparked) 94.88533163
(B, 4, unoccupied, parked) 60.99000000
(B, 4, occupied, unparked) 98.98010000
(B, 4, occupied, parked) 100.99000000
(B, 5, unoccupied, unparked) 93.66452584
(B, 5, unoccupied, parked) 50.99000000
(B, 5, occupied, unparked) 98.98010000
(B, 5, occupied, parked) 100.99000000
(B, 6, unoccupied, unparked) 92.26136635
(B, 6, unoccupied, parked) 40.99000000
(B, 6, occupied, unparked) 98.98010000
(B, 6, occupied, parked) 100.99000000
(B, 7, unoccupied, unparked) 91.01893392
(B, 7, unoccupied, parked) 30.99000000
(B, 7, occupied, unparked) 98.98010000
(B, 7, occupied, parked) 100.99000000
(B, 8, unoccupied, unparked) 90.36341158
(B, 8, unoccupied, parked) 20.99000000
(B, 8, occupied, unparked) 98.98010000
(B, 8, occupied, parked) 100.99000000
(B, 9, unoccupied, unparked) 90.61281790
(B, 9, unoccupied, parked) 10.99000000
(B, 9, occupied, unparked) 98.98010000
(B, 9, occupied, parked) 100.99000000
(terminal state) 1.00000000
```

The value function does not have negative rewards anymore. The states with the highest value (100.99) are the states when the parking spot is occupied and the agent is parked. So the optimal states are those when the agent is parked with a collision? This may not make sense at first but it's plausible given the policy below.

Here is the policy for the reckless driver:

```
(A, 0, unoccupied, unparked) PARK
```

```
(A, 0, unoccupied, parked) EXIT
(A, 0, occupied, unparked) PARK
(A, 0, occupied, parked) EXIT
(A, 1, unoccupied, unparked) DRIVE
(A, 1, unoccupied, parked) EXIT
(A, 1, occupied, unparked) PARK
(A, 1, occupied, parked) EXIT
(A, 2, unoccupied, unparked) DRIVE
(A, 2, unoccupied, parked) EXIT
(A, 2, occupied, unparked) PARK
(A, 2, occupied, parked) EXIT
(A, 3, unoccupied, unparked) DRIVE
(A, 3, unoccupied, parked) EXIT
(A, 3, occupied, unparked) PARK
(A, 3, occupied, parked) EXIT
(A, 4, unoccupied, unparked) DRIVE
(A, 4, unoccupied, parked) EXIT
(A, 4, occupied, unparked) PARK
(A, 4, occupied, parked) EXIT
(A, 5, unoccupied, unparked) DRIVE
(A, 5, unoccupied, parked) EXIT
(A, 5, occupied, unparked) PARK
(A, 5, occupied, parked) EXIT
(A, 6, unoccupied, unparked) DRIVE
(A, 6, unoccupied, parked) EXIT
(A, 6, occupied, unparked) PARK
(A, 6, occupied, parked) EXIT
(A, 7, unoccupied, unparked) DRIVE
(A, 7, unoccupied, parked) EXIT
(A, 7, occupied, unparked) PARK
(A, 7, occupied, parked) EXIT
(A, 8, unoccupied, unparked) DRIVE
(A, 8, unoccupied, parked) EXIT
(A, 8, occupied, unparked) PARK
(A, 8, occupied, parked) EXIT
(A, 9, unoccupied, unparked) DRIVE
(A, 9, unoccupied, parked) EXIT
(A, 9, occupied, unparked) PARK
(A, 9, occupied, parked) EXIT
(B, 0, unoccupied, unparked) PARK
(B, 0, unoccupied, parked) EXIT
(B, 0, occupied, unparked) PARK
(B, 0, occupied, parked) EXIT
(B, 1, unoccupied, unparked) DRIVE
```

```
(B, 1, unoccupied, parked) EXIT
(B, 1, occupied, unparked) PARK
(B, 1, occupied, parked) EXIT
(B, 2, unoccupied, unparked) DRIVE
(B, 2, unoccupied, parked) EXIT
(B, 2, occupied, unparked) PARK
(B, 2, occupied, parked) EXIT
(B, 3, unoccupied, unparked) DRIVE
(B, 3, unoccupied, parked) EXIT
(B, 3, occupied, unparked) PARK
(B, 3, occupied, parked) EXIT
(B, 4, unoccupied, unparked) DRIVE
(B, 4, unoccupied, parked) EXIT
(B, 4, occupied, unparked) PARK
(B, 4, occupied, parked) EXIT
(B, 5, unoccupied, unparked) DRIVE
(B, 5, unoccupied, parked) EXIT
(B, 5, occupied, unparked) PARK
(B, 5, occupied, parked) EXIT
(B, 6, unoccupied, unparked) DRIVE
(B, 6, unoccupied, parked) EXIT
(B, 6, occupied, unparked) PARK
(B, 6, occupied, parked) EXIT
(B, 7, unoccupied, unparked) DRIVE
(B, 7, unoccupied, parked) EXIT
(B, 7, occupied, unparked) PARK
(B, 7, occupied, parked) EXIT
(B, 8, unoccupied, unparked) DRIVE
(B, 8, unoccupied, parked) EXIT
(B, 8, occupied, unparked) PARK
(B, 8, occupied, parked) EXIT
(B, 9, unoccupied, unparked) DRIVE
(B, 9, unoccupied, parked) EXIT
(B, 9, occupied, unparked) PARK
(B, 9, occupied, parked) EXIT
(terminal state) N/A
```

Some aspects of the policy makes sense. When parked, the next action is EXIT. When not parked, the action is either DRIVE or PARK.

What's interesting is the decision to DRIVE or PARK. The optimal action at a handicap spot is now to PARK. This makes sense because the reward is high enough so that it is desirable to park. Interestingly, when a spot is occupied, the optimal action is to PARK! There is a high enough reward to cause a collision so this is the optimal action. This is the behavior of a reckless driver.

What is an interesting side effect is that the only unoccupied spots where PARK is the optimal action is A0 and B0. A1, B1, B2 and B3 are no longer unoccupied spots where PARK is optimal! A plausible explanation is that the agent is maximizing the chances of causing a collision (since there's high reward for collisions) so it keeps on driving around the parking lot until there is a occupied spot to cause a collision. This was an interesting consequence that we did not expect at first. In conclusion, this is clearly a reckless driver.

3.4 What Next

Variation: If, instead of the reckless driver, we simply take the parameters of the reasonable driver and only change the handicap reward to something high, then we see that the optimal policy is the same as that of the reasonable driver, except that the optimal action is to park when the agent is at a handicap spot that is unoccupied. One can confirm this using the code. It just sounded more fun to add a collision reward and see what happens.

Please feel free to modify the parameters of the parking MDP to explore other settings. The constructor of the ParkingMDP class has options to set.