Dynamic Programming and Reinforcement Learning Assignment4 Report

Chih-Chieh Lin (2700266)
Jiacheng Lu (2686004)

January 3, 2021

Introduction

This Report describes our solutions and methods for solving a shortest route problem in a 50x50 crossings system. Regarding the several questions mentioned in this assignment, the relevant solutions will be placed in the following sections.

1 Realization of Congestion

To realize the transition probability, we create an three-dimensional array composed of the probability to four direction in each state. To be more specific, the size of this array is 50x50x4. Take [1,1,0] for example, it represents the probability of successfully reaching the state on top of [1,1]. We define that 0,1,2,3 represent the probability of going up, down, left, right. Every road segment transition probability is generated randomly from 0.1, ..., 1.

2 Simple Heuristic

As for simple heuristic method, we create another array composed of 1 dividing by the probability. We consider the 1/probability as the cost when the agent decide to pass through the road. Afterwards, we implement the simple heuristic method by this array. This method only takes the roads with the direction pointing to target coordinate, and always picks the road with minimum cost.

3 System of Equations

In this section, we managed to use Bellman-Ford algorithm to find the shortest path to target coordinate(0,9). The value in the array composed of 1 dividing by probability is considered as the weights, so that we can use these weights to find the shortest path to the target.

$$weight = \frac{1}{transition\_probability}$$

The equations for Bellman-Ford is:

$$cost[v] = min(cost[u] + cost(v, u))$$

Therefore, this method would provide us with the minimum cost from given state to the target state; and it's much smaller than the one obtained by heuristic method. Part of the final minimum cost matrix is shown in appendix 1

4 Dynamic Programming

For dynamic programming, here we apply value iteration. We compute the optimal state value function by iteratively improving the estimate of $V(s)$. To simplify the calculation, we set the reward(time cost) at -1 for each action. The function terminates when reaching target(0,9). Firstly, for each state s, we initialize $V(s) = 0$. Secondly, for each state, we update:

$$V(s) = R(s) + max_{a \in A} \gamma * \sum_{s'} P_{sa}(s')V(s')$$

We repeat step two until convergence. Here we set $\epsilon = 0.0001$, it takes 160 iterations to converge. Then we can have the fastest route and lowest cost of reaching target. The expected cost matrix is shown in Figure2.

5 Q-learning with ε-greedy

When it comes to Q-learning, we consider the following equation:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot maxQ(s_{t+1}, a) - Q(s_t, a_t)]$$

We create an 50*50*4 Q-table, 4 represents four potential actions.

1. Initialize the Q-value to 0
2. Choose an action to execute according to the Q-values
3. Update Q-table

We repeat step 23 until reaching target and get an reward of 1. For ϵ-greedy, we set $learning\_rate = 0.1$, $\epsilon = 0.99$, $\epsilon\_decay = 0.005$. We set start point at (49, 49) and run 2500 episodes to test the performance, the result shows very clearly that the cost reduced from 15566 in the first episode to 162 in the last episode. The converging procedure was shown in Figure 5 For random start point, we test 100000 episodes. Part of the Q-table is shown in Figure 3

6 Modification of Reward Function

In this section, we modify the reward function to ease learning with Q-learning while introducing a limited bias. We decided to add reward while ta king actions that intended to move towards destination. For example, when the current state is at the left side of (0, 9), and the action is to move right, there is a direction reward parameter in the reward function. We take $directionReward = 0.05$ Then the equation for actions moving towards target is transformed to:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r + directionReward + \gamma \cdot maxQ(s_{t+1}, a) - Q(s_t, a_t)]$$

The rest steps is similar to the above section. We set 10000 episodes from random start points. In order to avoid the situation of stuck in one episode, we set the maximum step for each episode at 5000. We find out that the running time of each episode is much longer than Q-learning without bias, and much frequent of quitting episode because of max step restriction. The Q-table is shown in 4. We find out that although Q-values have variants, the best action (action with the biggest Q-value) is close to last section.

7 Comparison of Different Approaches

The result of value iteration and system of equation is very close. For example, starting from (49, 49) the minimum expect cost is 115.99 for dynamic programming and 115.73 for system of equations. The result of a simple heuristic algorithm is 156.17. The heuristic runs pretty fast in our test, while dynamic programming and system of equation take more time. At our parameter set for Q-learning, the minimum cost for starting from (49, 49) is 162. Furthermore, we show the movement procedure in Figure 6.

We also tried different parameter set while implementing Q-learning, and find out that an appropriate parameter setting impacts running time greatly. Due to the restriction of episodes, some of the Q-table is not updated and remained 0. However, for those updated points, the best action is close in bias and non-bias Q-table.

Appendix A Results

	0	1	2	3	4	5	6	7	8	9	10	11
0	14.56349	13.13492	10.63492	8.96825	7.53968	6.22222	4.22222	2.22222	1.11111	0.00000	5.00000	6.42857
1	14.56349	13.13492	10.63492	7.53968	6.42857	5.00000	3.88889	2.22222	1.11111	3.33333	2.50000	6.42857
2	15.37302	12.03968	9.53968	8.53968	11.42857	6.42857	5.31746	3.33333	2.36111	3.33333	4.16667	6.66667
3	16.53968	13.20635	11.20635	9.96825	11.07937	7.98413	6.98413	5.33333	3.47222	4.33333	5.41667	7.41667
4	15.88492	14.45635	12.50794	11.07937	11.32937	9.66270	8.23413	7.00000	5.97222	8.47222	6.52778	9.86111
5	17.13492	15.04762	13.93651	14.42460	12.99603	11.15079	9.48413	8.75000	7.63889	9.30556	7.77778	13.19444
6	19.13492	17.54762	18.93651	20.15079	14.66270	12.15079	11.49603	10.06746	9.06746	10.55556	9.44444	14.19444
7	20.13492	19.48413	18.48413	16.81746	15.81746	13.81746	13.49603	11.06746	10.49603	11.94444	10.69444	15.30556
8	21.56349	20.42857	19.17857	17.92857	16.49603	15.24603	13.56746	12.31746	13.56746	13.61111	11.69444	16.69444
9	22.15079	21.15079	20.03968	19.03968	18.17857	15.67857	14.56746	13.98413	15.98413	17.65079	14.19444	16.19444
10	24.15079	22.15079	21.46825	20.71825	18.21825	17.10714	17.41270	15.98413	16.98413	19.31746	15.44444	16.44444
11	24.91270	23.24603	22.13492	20.46825	19.21825	18.10714	19.41270	17.09524	18.09524	19.34524	18.77778	18.44444
12	26.99603	24.49603	25.46825	25.88492	22.55159	21.44048	20.84127	18.34524	20.34524	20.59524	20.20635	22.44444
13	26.92460	25.92460	26.89683	25.32937	23.66270	22.69048	22.50794	19.59524	23.27381	21.84524	21.20635	22.48413
14	30.25794	28.42460	28.18651	26.75794	25.09127	26.20238	27.86905	22.92857	24.52381	25.17857	23.20635	24.28968
15	30.78571	29.67460	29.42460	28.42460	29.67460	27.20238	27.51190	24.17857	25.84524	26.12302	24.45635	24.87302
16	35.78571	31.34127	30.53571	30.42460	32.17460	33.42460	26.85714	25.60714	28.10714	27.55159	28.80159	28.20635
17	36.34127	33.00794	34.25794	33.60714	31.94048	30.27381	28.27381	27.27381	28.70238	29.95238	30.46825	29.63492
18	38.00794	34.67460	35.05159	34.05159	33.05159	31.05159	29.38492	29.77381	32.03571	31.06349	36.06349	31.06349
19	39.17460	36.67460	37.16270	35.16270	36.27381	35.81349	30.81349	30.88492	31.99603	32.06349	34.06349	32.49206

Figure 1: Part of the expected cost Matrix calculated by system of equation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	15.7738	14.5238	13.0952	10.5952	8.92857	7.61111	5.11111	3.11111	1.11111	0	1.42857	2.85714	7.52381	8.77381	10.0
1	17.4405	16.2738	14.2738	11.5952	7.67857	6.56746	5.13889	4.02778	2.36111	3.33333	3.09524	4.09524	5.52381	8.85714	11.3
2	19.5119	16.1786	12.8452	10.3452	9.34524	8.23413	6.58333	5.47222	3.47222	4.33333	4.34524	6.09524	8.02381	10.2857	12.2
3	18.0397	16.7897	13.4563	11.4563	11.3452	10.2341	7.83333	7.13889	5.97222	7.12302	5.45635	9.42857	11.4286	11.9524	13.3
4	17.5635	16.3135	14.8849	14.4405	13.0119	10.75	9.08333	9.63889	7.63889	8.23413	6.70635	7.81746	8.81746	11.3175	12.4
5	19.5635	18.8135	15.4008	14.2897	13.1786	11.75	11.8452	10.1786	9.06746	9.48413	8.37302	8.81746	10.0675	12.5675	14.2
6	20.5635	22.7341	18.623	15.2897	16.5119	13.4167	12.4286	11.1786	10.496	10.9127	9.62302	9.92857	12.5675	13.5675	14.6
7	21.9921	19.4008	17.4008	16.4008	15.8452	14.8452	13.8571	12.4286	13.8294	12.5794	10.623	16.373	15.0675	14.8175	15.7
8	21.9008	20.9008	18.4008	17.5119	18.6071	16.1071	14.8571	14.0952	16.5952	14.7897	13.123	15.123	16.7897	16.8175	17.0
9	23.9008	20.8294	19.8294	20.2024	19.2024	17.5357	17.2063	16.0952	17.1508	16.0397	14.373	16.2341	17.4841	18.7341	20.1
10	24.3294	23.3294	21.2579	21.2024	20.2024	18.5357	19.2063	17.2063	18.2619	17.4683	17.7063	18.2341	18.4841	19.7341	21.5
11	26.5794	24.5794	24.5913	24.0397	22.373	21.123	19.4563	18.4563	19.7063	18.7183	19.1349	20.1349	19.7341	21.1627	24.4
12	28.246	26.0079	26.0198	25.1508	23.4841	22.373	21.123	19.7063	20.8175	19.9683	20.1349	21.3849	21.1627	22.1627	24.6
13	29.6984	28.4484	27.4484	26.8175	24.9127	25.7183	24.2897	23.0397	22.0675	23.3016	22.1349	23.3849	22.4127	23.4127	24.4
14	31.127	29.6984	29.4484	28.4841	28.3849	26.7183	25.5397	24.2897	25.4008	24.4127	23.3849	25.0516	23.4127	24.6627	26.9
15	32.4762	31.3651	30.5595	30.4841	30.3294	29.2183	26.7183	25.7183	26.9683	25.8413	26.8413	32.9524	24.6627	25.6627	28.1
16	34.4603	33.0317	35.5317	33.6468	32.2183	31.2183	31.7183	27.3849	32.3849	31.8413	28.5079	29.619	30.7302	30.6627	29.8
17	36.4603	34.6984	38.0317	38.1627	33.3294	32.8849	31.8849	29.8849	31.1349	32.1349	32.2976	31.0476	32.7143	31.7738	32.3
18	37.6984	36.6984	39.1984	37.1627	36.1627	35.1627	32.6627	30.996	32.3849	33.1349	33.7262	32.4762	34.4762	34.2738	35.6
19	38.6984	39.6984	40.5913	39.1627	37.4127	36.4127	37.5238	34.8849	33.6349	34.5635	35.9921	35.8095	36.6349	35.3849	38.1
20	40.127	43.4603	43.0913	41.1627	39.5079	37.8413	37.6627	35.996	36.1349	36.2302	38.2302	38.3095	37.9246	36.496	40.1
21	41.377	43.377	44.4881	45.3968	40.381	38.9524	38.7738	40.4405	37.1349	38.1349	39.3413	39.3095	39.1627	38.1627	41.2
22	43.877	44.8056	45.3968	43.7302	42.0635	40.9524	41.4206	39.9921	38.5635	40.1349	43.4683	40.4206	40.4127	39.4127	40.5

Figure 2: Part of the expected cost Matrix calculated by dynamical programming

	0	1	2	3
0	0.00000	0.00000	0.00000	0.00039
1	0.00000	0.00909	0.00000	0.00000
2	0.00000	0.00000	0.00117	0.00000
3	0.00000	0.00000	0.00009	0.00000
4	0.00000	0.00000	0.00000	0.00000
5	0.00000	0.08046	0.00000	0.00060
6	0.00000	0.01216	0.00699	0.37620
7	0.00000	0.08316	0.26429	0.97529
8	0.00000	0.29730	0.30107	0.99866
9	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.97823	0.99499	0.93936
11	0.00000	0.91271	0.16014	0.11928
12	0.00000	0.93549	0.07229	0.02976
13	0.00000	0.00317	0.90420	0.03357
14	0.00000	0.00918	0.79512	0.00064
15	0.00000	0.00002	0.56917	0.00876
16	0.00000	0.00512	0.29692	0.00001
17	0.00000	0.00855	0.10498	0.00081
18	0.00000	0.00003	0.01680	0.00002
19	0.00000	0.00000	0.00032	0.12974

Figure 3: Part of the Q-table(first line) Q-learning with ε-greedy

	0	1	2	3
0	0.00000	0.00001	0.00000	0.00250
1	0.00000	0.00000	0.00003	0.01150
2	0.00000	0.00164	0.00020	0.04332
3	0.00000	0.00286	0.00107	0.14541
4	0.00000	0.05114	0.00666	0.38766
5	0.00000	0.10491	0.09363	0.74757
6	0.00000	0.87864	0.89518	0.95774
7	0.00000	0.91684	0.94595	0.97980
8	0.00000	0.87838	0.78400	0.99882
9	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.61702	0.99604	0.46470
11	0.00000	0.00000	0.22192	0.00895
12	0.00000	0.91744	0.00001	0.01489
13	0.00000	0.00541	0.86486	0.02529
14	0.00000	0.00629	0.77561	0.00477
15	0.00000	0.00030	0.64871	0.00000
16	0.00000	0.00019	0.05906	0.00001
17	0.00000	0.00000	0.00380	0.00000
18	0.00000	0.00000	0.00017	0.00000
19	0.00000	0.00000	0.00001	0.00000

Figure 4: Part of the Q-table(first line) Q-learning with limited bias

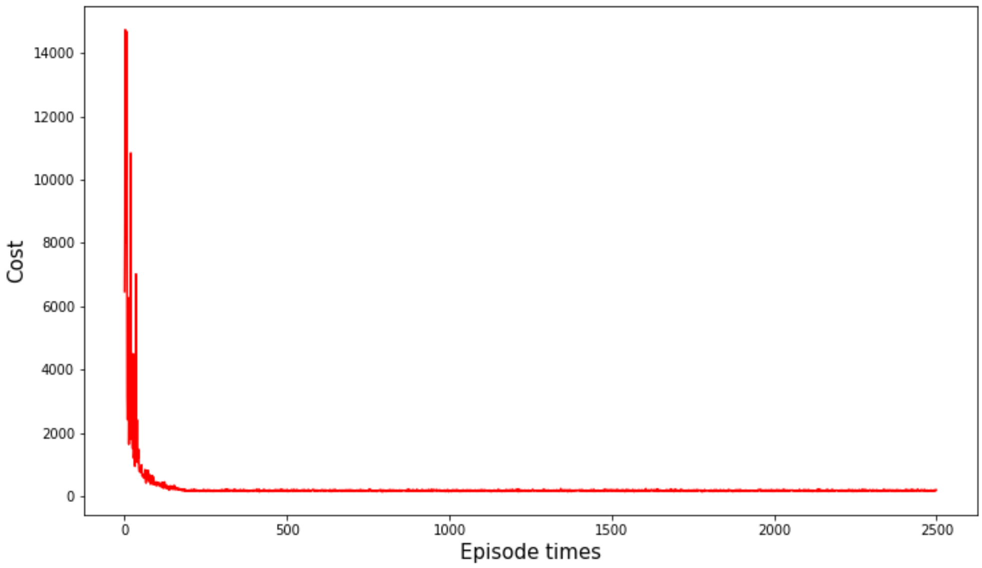

Figure 5: The Converging procedure of Q learning

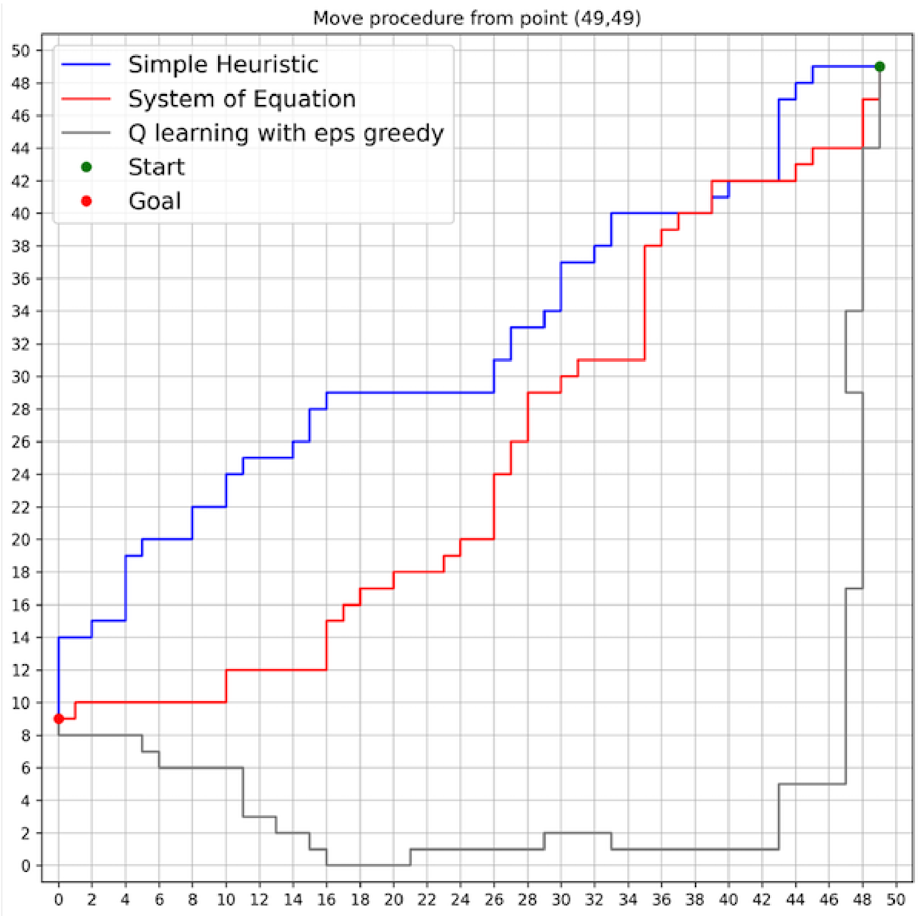

Figure 6: Move Procedure of 3 different methods