Autonomous Agents
Report Assignment 1: Single Agent Planning

Authors:

Agnes van Belle *(10363130)*,
Maaike Fleuren *(10350470)*,
Norbert Heijne *(10357769)*,
Lydia Mennes *(10333843)*

September 21, 2012

1 Introduction

This report has been written for the Master Artificial Intelligence course Autonomous Agents. This report will contain the answers, motivation and explanation for our implementations of the tasks we had to accomplish in our first assignment for this course. These taks were centered around the topic of "Single Agent Planning".

1.1 The Environment

In all tasks there is assumed to be a grid world (of 11×11) with a predator and a prey in it. The agents can both move one tile forward each iteration. The direction they take (or if they move at all) is affected by probabilities (their policies). If they move over the edge of the grid they end up at the opposing side of the grid. The prey will never step into the predator. We are focused on improving the decisions of one agent, the predator.

2 Simulating the environment

A first task was to write a simulator for the environment as defined in Section 1.1.

The choice has been made to not encode the positions of the agents as part of a grid, i.e. matrix. Instead the agents both know their own position and on each iteration the position of the other agent is given as input by the environment, as we have stated in the Agent interface. This is needed for prey to see if the predator is next to it, to prevent it moving towards the prey. In the predator's case it might be necessary later on to know the position of the prey although it is not necessary for this particular sub-assignment.

2.1 Results

A mean and a standard deviation was asked for 100 runs with the use of the random policy for the predator's behaviour. For the exact output, one can consult Appendix A. The lowest amount of time steps observed was 19 time steps. The optimal amount of time steps given that the prey would remain still throughout the trial run would be 10. The highest amount observed was 1194 steps. The average amount of time steps was 296.93 time steps and the standard deviation was 244.55 time steps (rounded up).

3 Planning for the Environment

It is possible for the predator to develop a more sophisticated policy by planning. For this we need to assume that the agent has a full and accurate model of the environment. Then we can make use of several dynamic programming planning algorithms, and we will discuss our implementation of these in the subsections below.

Each of these algorithms makes use of a matrix or structure V wherein the values for all states are put. Each of these algorithms furthermore make use of two important parameters. The first, θ, specifies for which amount of change in the V-values the update in V-values terminates. The stopping criterion is given by $\max_{s \in \mathcal{S}} |V_{k+1}(s) - V_k(s)| < \theta$. The second, γ, is the discount factor and specifies how much emphasis should be placed on previous values for V.

3.1 Policy Evaluation

The algorithm for Policy Evaluation follows from turning the Bellman equation into an update rule, such that in each iteration of Policy Evaluation, Equation 1 is used for every $s \in \mathcal{S}^+$ (where $\mathcal{S}^+$ denotes the set of states including the terminal state).

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s')\right) \tag{1}$$

Because we used the random policy for the predator, and there were five actions possible for the predator, Equation 1 could be simplified to Equation 2.

$$V_{k+1}(s) \leftarrow \frac{1}{5} \sum_a \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s')\right) \tag{2}$$

Where $\mathcal{P}^a_{ss'}$ has to take into account the possible movements of the prey. Note that $\mathcal{R}^a_{ss'}$ was always zero, except for the single goal state.

We implemented Policy Evaluation using a 121×121 matrix to hold the V- values for the states.

3.1.1 Results

It took 111 iterations until the stopping criterion $\max_{s \in \mathcal{S}} |V_{k+1}(s) - V_k(s)| < \theta$ was met using $\theta = 0$ and $\gamma = 0.8$. In Table 2 we show some values that Policy Evaluation found for specific states.

To get an intuitive grasp of the resulting V-values, a colormap of all final V-values is given in Figure 1.

Predator position	Prey position	Value
(0,0)	**(5,5)**	0.0060
(2,3)	**(5,4)**	0.1820
(2,10)	**(10,0)**	0.1820
(10,10)	**(10,0)**	1.1950

Table 1: State values using Policy Evaluation ($\theta = 0$, $\gamma = 0.8$)

3.2 Policy Iteration

Of course the reason for computing the values of all states using Policy Evaluation is to find a good policy. Policy Evaluation gets fed a policy, then computes the values for all states until it converges (i.e. until the largest change in value in one sweep is below θ), and then stops. So after Policy Evaluation we could construct a new policy based on these new values, that would be just as good or better than the policy originally fed to Policy Evaluation. The method of Policy Iteration captures this idea, by repeatedly letting the method of Policy Evaluation be followed by another method called Policy Improvement, until the policy doesn't change anymore. The method of Policy Improvement is explained in Section 3.2.1.

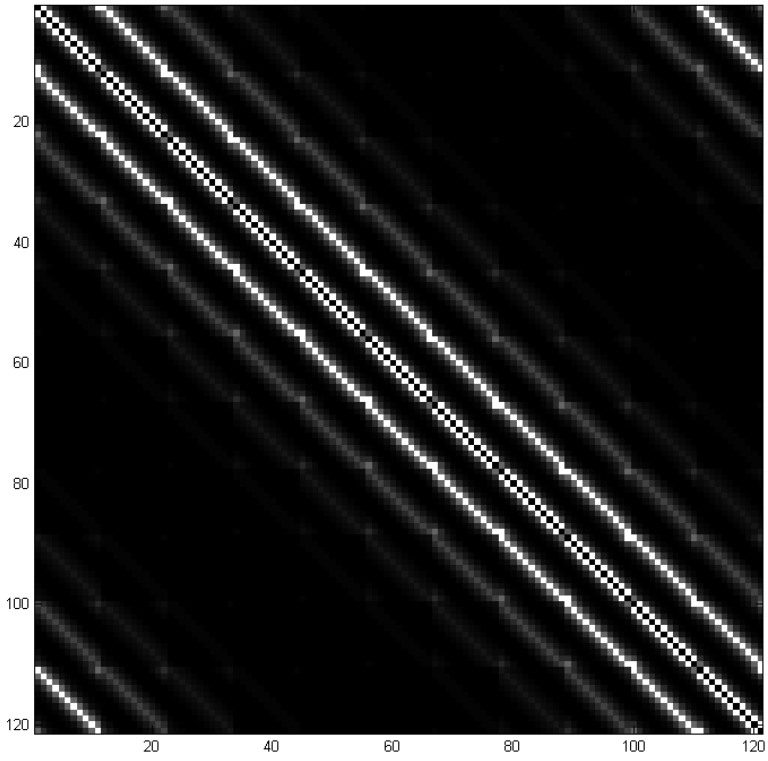

Figure 1: Colormap of the V-values resulting from Policy Evaluation for $\theta = 0$ and $\gamma = 0.8$.
Each axis represents all tiles of the grid world. The brighter the color the higher the corresponding V-value.

3.2.1 Policy Improvement

The idea behind Policy Improvement is that you have determined the value function V^π for a certain policy π, and want to know now if any change in the policy could yield a better expected return given this value function. Policy Improvement sweeps through all states to check if any action a would give a better expected return than the recommended action $\pi(s)$ at that state and if so, $\pi(s) \leftarrow a$. In our stochastic case, this means we need to check for a better probability distribution over the actions at each state, and use the update rule

$$\pi^{k+1}(s) \leftarrow \arg max_{\pi'(s)} \sum_a \pi'(s,a) Q^{\pi^k}(s,a) \tag{3}$$

Policy Improvement thus yields an improved policy, which can then used to compute the new value function V by Policy Evaluation. This process is repeated by Policy Iteration.

3.2.2 Results

The Policy Iteration algorithm has been implemented and run using different settings of γ. The used values are $\gamma = 0.1$, $\gamma = 0.5$, $\gamma = 0.7$ and $\gamma = 0.9$. For each of these runs θ is set to 0. In this report, we provide the V-values for the states in which the prey is located at **(5,5)**. The values we found by running Policy Iteration parameterised this way, can be found in Tables **??**, 4, 5 and 6. They are exactly the same as those we found with our implementation of Value Iteration (see Section 3.3).

As expected the V-values are higher near the goal state, which is **(5,5)** in this specific case. This will cause the predator to move towards the prey in all cases. Also the values further from the goal state decrease

faster in value for lower learning rates. The maximal V-value is 10, which is to be expected with a maximal reward of 10.

3.3 Value Iteration

Value Iteration is another planning algorithm. Just like Policy Iteration (Section 3.2), this algorithm combines the steps of Policy Evaluation and Policy Improvement. However, Value Iteration uses a different approach. Instead of letting Policy Evaluation run until it had converged, Policy Evaluation is stopped after one sweep. In that sweep it combines Policy Evaluation with Policy Improvement in each update step for every state. Its update rule for eahc state is given by Equation 4.

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s')\right) \tag{4}$$

The algorithm does so by replacing the value of the $(k+1)^{th}$ iteration with the expected reward of the action that maximizes this expectation (based on the V-value of s' of the k^{th} iteration and the immediate reward of s'), instead of a weighted sum of the expected rewards of all actions.

3.3.1 Results

The Value Iteration algorithm has also been implemented and run using different settings of γ. The used values are $\gamma = 0.1$, $\gamma = 0.5$, $\gamma = 0.7$ and $\gamma = 0.9$. For each of these runs θ is set to 0. In this report, we provide the V-values for the states in which the prey is located at **(5,5)**. The results can be found in Tables **??**, 4, 5 and 6. These results are exactly the same as those we found with Policy Iteration. A representation of the V-values which nicely shows the decrease of the V-values further from the goal state can be found in figure 2. Here, the symmetry of the state space becomes apparent.

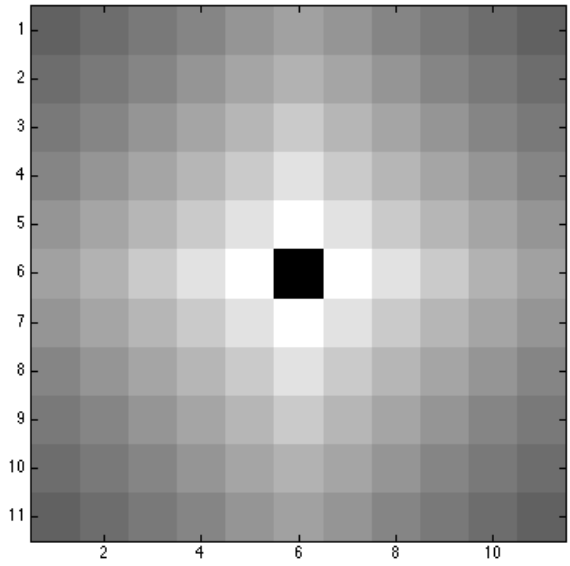

Figure 2: Colormap of the V-values resulting from Value Iteration for $\theta = 0$ and $\gamma = 0.9$. The brighter the color the higher the corresponding V-value.

The convergence speed in numbers of iterations is different from those found with Policy Iteration, however. We give an overview in Table 2.

However, note that each iteration of Policy Iteration involves a call to Policy Evaluation and Policy Improvement, and both Policy Evaluation and Policy Improvement will sweep through the whole state

	Iterations V.I.	Iterations P.I.
$\gamma = 0.1$	20	8
$\gamma = 0.5$	28	7
$\gamma = 0.7$	31	7
$\gamma = 0.9$	34	8

Table 2: Comparison of iterations of Value Iteration and Policy Iteration

space. Policy Improvement will do this one time, but Policy Evaluation will often do this multiple times (until the largest change of value is below the threshold θ). Therefore one would be mistaken to compare these two algorithms on the number of their largest-level iterations used as an indication of computational complexity.

Note, too, that it would be a mistake to compare between the number of iterations for differently parameterized Policy Iteration algorithms. For example, using $\gamma = 0.1$ we need the same number of iterations as when using $\gamma = 0.9$ for convergence, however, comparing the number of times that Policy Evaluation is invoked under these settings, we saw that using the setting of $\gamma = 0.9$ Policy Iteration had to invoke it approximately 2.45 times as much as when using the setting of $\gamma = 0.1$.

4 (SC) State space reduction

In the experiments described in all following sections, unless stated otherwise, we used a state space that ...
(uitleg default state space)
We will refer to this state space as the "default" state space.

We elaborated on a more efficient representation for the states, and finally contrived one consisting of 30 states, an approximately 488 times smaller one than the default state space. We will refer to this state space as the "efficient"' state space. This state space...
(uitleg efficiente state space)

5 Conclusion

Bla bla

Appendices

A Simulating the Environment

Our program's output for the first 100 runs for the random policy predator.

```
Timesteps:421
Timesteps:831
Timesteps:476
Timesteps:74
Timesteps:537
Timesteps:40
Timesteps:468
Timesteps:465
Timesteps:105
Timesteps:123
Timesteps:227
Timesteps:658
Timesteps:696
Timesteps:153
Timesteps:426
Timesteps:431
Timesteps:24
Timesteps:197
Timesteps:517
Timesteps:313
Timesteps:492
Timesteps:213
Timesteps:457
Timesteps:392
Timesteps:47
Timesteps:178
Timesteps:459
Timesteps:624
Timesteps:881
Timesteps:100
Timesteps:244
Timesteps:127
Timesteps:213
Timesteps:145
Timesteps:45
Timesteps:301
Timesteps:628
Timesteps:248
Timesteps:88
Timesteps:123
Timesteps:82
Timesteps:206
Timesteps:181
Timesteps:771
Timesteps:114
Timesteps:238
Timesteps:118
Timesteps:67
Timesteps:41
Timesteps:662
Timesteps:27
Timesteps:73
Timesteps:217
Timesteps:269
Timesteps:382
Timesteps:60
Timesteps:205
Timesteps:64
Timesteps:133
Timesteps:232
Timesteps:148
Timesteps:504
Timesteps:113
Timesteps:316
Timesteps:151
Timesteps:178
Timesteps:53
Timesteps:526
Timesteps:150
Timesteps:690
Timesteps:490
```

```
Timesteps:116
Timesteps:288
Timesteps:79
Timesteps:163
Timesteps:266
Timesteps:566
Timesteps:1194
Timesteps:133
Timesteps:690
Timesteps:136
Timesteps:121
Timesteps:123
Timesteps:492
Timesteps:288
Timesteps:185
Timesteps:19
Timesteps:78
Timesteps:250
Timesteps:42
Timesteps:268
Timesteps:190
Timesteps:231
Timesteps:393
Timesteps:338
Timesteps:100
Timesteps:49
Timesteps:653
Timesteps:1173
Timesteps:421
Average timesteps over 100 trials: 296.93
Standard deviation over 100 trials: 244.54689754728025
```

B Policy Iteration and Value Iteration results

	0	1	2	3	4	5	6	7	8	9	10
0	0.000000	0.000002	0.000011	0.000074	0.000438	0.001730	0.000438	0.000074	0.000011	0.000002	0.000000
1	0.000002	0.000011	0.000075	0.000498	0.003195	0.013773	0.003195	0.000498	0.000075	0.000011	0.000002
2	0.000011	0.000075	0.000498	0.003443	0.021739	0.117564	0.021739	0.003443	0.000498	0.000075	0.000011
3	0.000074	0.000498	0.003443	0.021739	0.166976	0.816327	0.166976	0.021739	0.003443	0.000498	0.000074
4	0.000438	0.003195	0.021739	0.166976	0.816327	10.000000	0.816327	0.166976	0.021739	0.003195	0.000438
5	0.001730	0.013773	0.117564	0.816327	10.000000	0.000000	10.000000	0.816327	0.117564	0.013773	0.001730
6	0.000438	0.003195	0.021739	0.166976	0.816327	10.000000	0.816327	0.166976	0.021739	0.003195	0.000438
7	0.000074	0.000498	0.003443	0.021739	0.166976	0.816327	0.166976	0.021739	0.003443	0.000498	0.000074
8	0.000011	0.000075	0.000498	0.003443	0.021739	0.117564	0.021739	0.003443	0.000498	0.000075	0.000011
9	0.000002	0.000011	0.000075	0.000498	0.003195	0.013773	0.003195	0.000498	0.000075	0.000011	0.000002
10	0.000000	0.000002	0.000011	0.000074	0.000438	0.001730	0.000438	0.000074	0.000011	0.000002	0.000000

Table 3: The V-values for Value Iteration, with $\gamma = 0.1$ and the prey at position (5,5). The convergence speed is 20 iterations.

	0	1	2	3	4	5	6	7	8	9	10
0	0.0267	0.0502	0.0943	0.1768	0.3328	0.5332	0.3328	0.1768	0.0943	0.0502	0.0267
1	0.0502	0.0924	0.1769	0.3390	0.6448	1.0814	0.6448	0.3390	0.1769	0.0924	0.0502
2	0.0943	0.1769	0.3390	0.6498	1.2435	2.2027	1.2435	0.6498	0.3390	0.1769	0.0943
3	0.1768	0.3390	0.6498	1.2435	2.3977	4.4444	2.3977	1.2435	0.6498	0.3390	0.1768
4	0.3328	0.6448	1.2435	2.3977	4.4444	10.0000	4.4444	2.3977	1.2435	0.6448	0.3328
5	0.5332	1.0814	2.2027	4.4444	10.0000	0.0000	10.0000	4.4444	2.2027	1.0814	0.5332
6	0.3328	0.6448	1.2435	2.3977	4.4444	10.0000	4.4444	2.3977	1.2435	0.6448	0.3328
7	0.1768	0.3390	0.6498	1.2435	2.3977	4.4444	2.3977	1.2435	0.6498	0.3390	0.1768
8	0.0943	0.1769	0.3390	0.6498	1.2435	2.2027	1.2435	0.6498	0.3390	0.1769	0.0943
9	0.0502	0.0924	0.1769	0.3390	0.6448	1.0814	0.6448	0.3390	0.1769	0.0924	0.0502
10	0.0267	0.0502	0.0943	0.1768	0.3328	0.5332	0.3328	0.1768	0.0943	0.0502	0.0267

Table 4: The V-values for Value Iteration, with $\gamma = 0.5$ and the prey at position (5,5). The convergence speed is 28 iterations.

	0	1	2	3	4	5	6	7	8	9	10
0	0.4348	0.6065	0.8453	1.1765	1.6463	2.1169	1.6463	1.1765	0.8453	0.6065	0.4348
1	0.6065	0.8328	1.1750	1.6587	2.3345	3.0759	2.3345	1.6587	1.1750	0.8328	0.6065
2	0.8453	1.1750	1.6587	2.3415	3.3044	4.4805	3.3044	2.3415	1.6587	1.1750	0.8453
3	1.1765	1.6587	2.3415	3.3044	4.6737	6.5116	4.6737	3.3044	2.3415	1.6587	1.1765
4	1.6463	2.3345	3.3044	4.6737	6.5116	10.0000	6.5116	4.6737	3.3044	2.3345	1.6463
5	2.1169	3.0759	4.4805	6.5116	10.0000	0.0000	10.0000	6.5116	4.4805	3.0759	2.1169
6	1.6463	2.3345	3.3044	4.6737	6.5116	10.0000	6.5116	4.6737	3.3044	2.3345	1.6463
7	1.1765	1.6587	2.3415	3.3044	4.6737	6.5116	4.6737	3.3044	2.3415	1.6587	1.1765
8	0.8453	1.1750	1.6587	2.3415	3.3044	4.4805	3.3044	2.3415	1.6587	1.1750	0.8453
9	0.6065	0.8328	1.1750	1.6587	2.3345	3.0759	2.3345	1.6587	1.1750	0.8328	0.6065
10	0.4348	0.6065	0.8453	1.1765	1.6463	2.1169	1.6463	1.1765	0.8453	0.6065	0.4348

Table 5: The V-values for Value Iteration, with $\gamma = 0.7$ and the prey at position (5,5). The convergence speed is 31 iterations.

	0	1	2	3	4	5	6	7	8	9	10
0	3.8831	4.2915	4.7417	5.2374	5.7919	6.2513	5.7919	5.2374	4.7417	4.2915	3.8831
1	4.2915	4.7118	5.2281	5.8024	6.4356	6.9973	6.4356	5.8024	5.2281	4.7118	4.2915
2	4.7417	5.2281	5.8024	6.4401	7.1476	7.8390	7.1476	6.4401	5.8024	5.2281	4.7417
3	5.2374	5.8024	6.4401	7.1476	7.9362	8.7805	7.9362	7.1476	6.4401	5.8024	5.2374
4	5.7919	6.4356	7.1476	7.9362	8.7805	10.0000	8.7805	7.9362	7.1476	6.4356	5.7919
5	6.2513	6.9973	7.8390	8.7805	10.0000	0.0000	10.0000	8.7805	7.8390	6.9973	6.2513
6	5.7919	6.4356	7.1476	7.9362	8.7805	10.0000	8.7805	7.9362	7.1476	6.4356	5.7919
7	5.2374	5.8024	6.4401	7.1476	7.9362	8.7805	7.9362	7.1476	6.4401	5.8024	5.2374
8	4.7417	5.2281	5.8024	6.4401	7.1476	7.8390	7.1476	6.4401	5.8024	5.2281	4.7417
9	4.2915	4.7118	5.2281	5.8024	6.4356	6.9973	6.4356	5.8024	5.2281	4.7118	4.2915
10	3.8831	4.2915	4.7417	5.2374	5.7919	6.2513	5.7919	5.2374	4.7417	4.2915	3.8831

Table 6: The V-values for Value Iteration, with $\gamma = 0.9$ and the prey at position (5,5). The convergence speed is 34 iterations.