Autonomous Agents
Assignment 1: *Single Agent Planning*

Artur Alkaim –
Peter Dekker – 10820973
Rafael Reia –
Yikang Wang – 10540288

November 14, 2014

1 Introduction

In this assignment, we study the application of dynamic programming algorithms to find a best policy for Markov Decision Processes (MDP's). The MDP we implement is the *predator versus prey grid world*. A predator and a prey move around a field, in which the predator wants to catch the prey. Our goal is to find the best poliy for the predator to catch the prey. We evaluate the following methods:

- a random policy
- Policy Evaluation
- Policy Iteration
- Value Iteration

2 Program design

Our program has been written in Java. The algorithm for every policy can be started from the corresponding main class: *Main*, *MainPE*, *MainPI* and *MainVI*. The main class creates an environment (specific for a certain policy), in which the predator and the prey reside. Then, it runs the simulation for that environment.

The grid location of the predator/prey is stored in the predator/prey object itself. There is no explicit grid object of which the predator and the prey are part. Using a subclass structure from *Predator*, there are different predator objects for the different policies. The prey always has the same policy: it stays on the same place with probability 0.8 and moves in one of the 4 directions with probability 0.05.

3 Algorithms

3.1 Random policy

Using the random policy, the predator moves in every direction (4 directions or stay on same place) with probability 0.2. Running the simulation 100 times and measuring the time it takes to catch the prey, we get the following results:

Run	Number of iterations	Run	Number of iterations
1	291	52	21
2	36	53	434
3	31	54	686
4	176	55	295
5	304	56	770
6	307	57	81
7	332	58	306
8	198	59	37
9	67	60	114
10	444	61	157
11	262	62	130
12	273	63	482
13	432	64	114
14	149	65	108
15	133	66	309
16	295	67	39
17	21	68	472
18	237	69	23
19	146	70	526
20	129	71	88
21	56	72	1210
22	173	73	337
23	841	74	404
24	173	75	141
25	899	76	111
26	66	77	375
27	398	78	27
28	291	79	116
29	32	80	58
30	31	81	396
31	37	82	1496
32	35	83	219
33	498	84	47
34	378	85	1493
35	366	86	363
36	144	87	301
37	143	88	215
38	155	89	49
39	700	90	50
40	503	91	91
41	510	92	91
42	103	93	181
43	826	94	160
44	100	95	57
45	143	96	208
46	104	97	1033
47	223	98	72
48	600	99	20
49	93	100	210
50	269	Average	279.45
51	69	Standard deviation	342.7141666170221

3.2 Policy evaluation

3.3 Policy iteration

3.4 Value iteration

4 Efficient state-space representation

The state-space consists of 11^4 items: both the predator and the prey have coordinates in a 11×11 space. It would save computation time if the policy algorithms do not have to loop over 11^4, but less, items.

A solution to this would be to represent the locations of the predator and the prey not as absolute locations, but as relative locations.

In this case, the predator stores a tuple $\delta_p(x, y)$ for every prey p, which denotes the distance to p. x and y are updated by +1 or -1 every time the predator or prey moves. Every iteration, the modulo of x and y is taken, because the environment is a torus:

$$x := x \, mod \, 11 \tag{1}$$

$$y := y \, mod \, 11 \tag{2}$$

5 Conclusion