Dynamic Programming & Reinforcement Learning 2020 **(Due: 29 nov 2020)**

Submission Assignment #2

Instructor: Ger Koole *Names:* Mick IJzer, *Netids:* 2552611

This assignment considers a system that slowly deteriorates over time. When it is new it has a failure probability of 0.1 and the probability of failure increases every time unit linearly with 0.01. When there is a failure the part has to be replaced. The replacement costs are 1, and after replacement the part is new (i.e. failure probability is 0.1 again). The probability of failure after t time units is given in equation 0.1. Obviously the probability of no failure is given by $1 - P_{failure}$. Given these equations it becomes clear that the age of the part cannot exceed 90 time units, since the probability of failure becomes 1.

$$P_{failure}(t) = 0.1 + 0.01t \tag{0.1}$$

1 Question A

To compute the stationary distribution of the system the balance equations have to be used. These are given in equations 1.1, 1.2. Equation 1.1 reflects the stationary distribution of π_0 and equation 1.2 describes the stationary distribution for all other states π_i. Equation 1.3 is the normalizing equation, where the sum of the probabilities of being in all possible states always sums up to 1. As stated earlier, the number of states is 91 (0 through 90), since after 90 time there will be a guaranteed failure.

$$\pi_0 = \sum_{i=1}^{90}(0.1 + 0.01i)\pi_i \tag{1.1}$$

$$\pi_i = (0.9 - 0.01i)\pi_{i-1} \qquad \forall \quad i \geq 1 \tag{1.2}$$

$$\sum_{i=0}^{90}\pi_i = 1 \tag{1.3}$$

The balance equations for π_i can be expressed in terms of π_0 as can be seen in equation 1.4. Next, the normalization equation 1.3 can be applied to obtain equation 1.5. A bit of rewriting results in the stationary distribution for π_0 as given in equation 1.6. Equation 1.6 can be solved numerically to find $\pi_0 = 0.146098$. The stationary distribution for the other states can be computed by using equation 1.4. Given the problem description π_0 is only visited after replacing a part. The act of replacing a part has costs $c = 1$ associated with it. Therefore, the long-run average replacement costs are $\pi_0 * c = 0.146098$.

$$\pi_i = \pi_0 * \prod_{j=0}^{i-1}(0.9 - 0.01j) \qquad \forall \quad i \geq 1 \tag{1.4}$$

$$\pi_0 + \pi_0 * \sum_{i=1}^{90}(\prod_{j=0}^{i-1} 0.9 - 0.01j) = 1 \tag{1.5}$$

$$\pi_0 = (1 + \sum_{i=1}^{90}(\prod_{j=0}^{i-1} 0.9 - 0.01j))^{-1} \tag{1.6}$$

2 Question B

The average-cost Poisson equation is first defined in equation 2.1. Note that $V(i+1)$ doesn't exist for $V(90)$. However this is not an issue, since the term in front of $V(i+1)$ becomes $(0.9 - 0.01 * 90) = 0$. Next, one value has to be set to a constant. In this case it was decided that $V(0) = 0$. This leads to equation 2.2, which can be rewritten to equation 2.3. This equation can be interpreted as a set of 91 linear equations, with 91 unknown

variables. This means that the matrix can be solved using a solver. Since ϕ is the long-term average reward, it is the value that should be extracted from the solution. The found value is $\phi = -0.146098$. However, since the question is based on costs it can be stated that the long-term average costs are $-\phi = 0.146098$, which corresponds to the costs found in section 1.

$$V(i) + \phi = (0.1 + 0.01i) * (V(0) - 1) + (0.9 - 0.01i) * V(i + 1) \tag{2.1}$$

$$V(i) + \phi = -(0.1 + 0.01i) + (0.9 - 0.01i) * V(i + 1) \tag{2.2}$$

$$V(i) - (0.9 - 0.01i) * V(i + 1) + \phi = -(0.1 + 0.01i) \tag{2.3}$$

3 Question C

In the previous questions there was only one single action that could be taken. For this question a possible action is added; preventive replacement. At any point in time the decision can be made to replace the part before it breaks. The costs associated with this action are 0.5, and after replacement the part is again as good as new. To find the optimal policy two approaches are used; policy iteration and value iteration.

Policy Iteration Policy iteration consists of two steps that are repeated, namely policy evaluation and policy improvement. The policy is evaluated by solving the Poisson equations in the same way as was done in section 2. However, the linear equations will change from equation 2.3 to 3.1 when the policy dictates preventive replacement (note that again $V(0) = 0$). When the solution to the linear equations are found, all possible actions in each state will be reevaluated after which the action with the highest expected value will be used in the new policy. This step is shown in equation 3.2. Now that the policy is evaluated and improved, the next iteration will start. Note that the initial policy doesn't matter.

$$V(i) + \phi = -0.5 \tag{3.1}$$

$$\pi_i(s) = \arg\max_{a \in A} \mathbb{E}(R(s, a, s') + V(s')) \tag{3.2}$$

Policy iteration was stopped when the policy before and after an iteration were equal. The optimal policy was found after 2 steps and it indicates that preventive replacement should always be done after 13 time units and later. Before that in the states 0 through 12 no action should be taken. Due to using the Poisson equations the long-term average costs can be easily extracted. Compared to the original long term average cost of $-\phi = 0.146098$, the new optimal policy results in a long term average cost of $-\phi = 0.14492$. The optimal policy, state values $V(s)$, and expected values $V(s) + \phi$ for all states can be found in table 1 in the Appendix. Note that action 1 corresponds to preventive replacement, and action 0 to no action.

Value Iteration Value iteration is similar to policy iteration in a way that there is an iterative process that reevaluates the value of the actions that can be taken in each state. However, in value iteration the policy doesn't have to be evaluated. For complex problems value iteration usually converges faster than policy iteration. However, value iteration is an approximation of the values of the states, while policy iteration can compute the exact expected values.

First, two containers are created where the values for each state $V(s)$ and the eventual policy can be stored. Then for each iteration a vector containing the "new" values $\widehat{V}$ is created, and filled by looping through each state and action. The actual value $\widehat{V}(s)$ for a state s is computed by equation 3.3. After each iteration the original container V is replaced by the vector of "new" values $\widehat{V}$ to be used in the next iteration. However, the "new" values do get their maximum value subtracted to keep the maximum value at 0 (i.e. $V = \widehat{V} - max(\widehat{V})$). The termination condition for the value iteration was that the maximum absolute difference between the values before and after the iteration was less than $1e^{-6}$. The value iteration was terminated after 64 steps. The found policy and state values are completely equal to the ones found in policy iteration (see table 1).

$$\widehat{V}(s) = \max_{a \in A} \mathbb{E}(R(s, a, s') + V(s')) \tag{3.3}$$

Appendix

State	**Action**	**V(s)**	**V(s) + ϕ**
0	0	-0.0000	-0.1449
1	0	-0.0499	-0.1948
2	0	-0.0953	-0.2402
3	0	-0.1366	-0.2815
4	0	-0.1741	-0.3191
5	0	-0.2083	-0.3532
6	0	-0.2390	-0.3840
7	0	-0.2667	-0.4115
8	0	-0.2910	-0.4359
9	0	-0.3121	-0.4570
10	0	-0.3296	-0.4746
11	0	-0.3432	-0.4881
12	0	-0.3521	-0.4970
13	1	-0.3551	-0.5000
≥ 14	1	-0.3551	-0.5000

Table 1: Optimal Policy and Expected Values