Description of Machine Replacement and River-swim tasks

March 2023

1 Machine Replacement Problem

The Machine Replacement problem is a classic optimization problem in operations research and production management. The problem involves determining the optimal time to replace a machine or equipment, such that the total costs associated with operating the machine are minimized.

The basic idea is that machines have a finite lifespan, and they eventually become less reliable and more expensive to maintain as they age. At some point, it becomes more cost-effective to replace the machine than to continue operating it. The goal of the Machine Replacement problem is to determine the optimal time to replace the machine, balancing the costs of operating the machine with the costs of replacing it.

The problem can be modeled as a dynamic programming problem, with the decision to replace the machine at each time step being based on the expected costs and benefits of doing so. The objective function is typically the total expected costs over the entire planning horizon, which includes both the costs of operating the machine and the costs of replacing it.

The Machine Replacement problem is relevant in a variety of industries, including manufacturing, transportation, and energy. It is important for companies to carefully manage the replacement of their equipment to avoid unexpected downtime and minimize costs. By solving the Machine Replacement problem, companies can make informed decisions about when to replace their machines, ensuring that they operate efficiently and cost-effectively.

The problem is of operating a machine effectively. The machine can be operated in one of n possible states($s = \{1, 2, \ldots, n\}$. Let state 1 be the perfect state.

Increasingly as move to higher states, the failure probability also increases that is $g(1) \leq g(2) \leq g(3) \ldots \leq g(n)$. Action space = Continue(C), Perform Maintenance(PM). The cost incurred for maintenance is R. Thus, the final cost incurred during maintenance is $R + g(1)$, where R is the repairing cost and $g(1)$

is the cost for functioning in state 1.

$$P_{i1}(PM) = 1 \tag{1}$$

$$P_{ij}(PM) = 0, \forall j \neq 1 \tag{2}$$

$$P_{ij}(C) = 0, \forall j < i \tag{3}$$

$$P_{ij}(C) \leq P_{(i+1)j}(C), \forall j > i \tag{4}$$

When the machine is operated in state j its well-being will deteriorate to another state $i \geq j$ after the current time period. The given optimal policy has a threshold optimal policy. If our objective is to minimize the cost of using the machine we use this structure to get speedy results. We should perform maintenance if and only if the state of machine $i \geq i^*$ such that i^* is a certain threshold state. Consider the Machine Replacement problem, introduced earlier, with four states the state space ($\mathcal{S}$) is defined as $\mathcal{S} = s_0, s_1, s_2, s_3$. Probability matrix ($\mathcal{P}$) and reward matrix ($\mathcal{R}$) are given as $\mathcal{P}[a_0] = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0 & 0.22 & 0.33 & 0.45 \\ 0 & 0 & 0.428 & 0.572 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\mathcal{P}[a_1] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ and $\mathcal{R}[a_0] = \begin{bmatrix} 0.1 & 0.397 & 0.693 & 0.99 \end{bmatrix}$, $\mathcal{R}[a_1] = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$. There are two actions in the action space ($\mathcal{A}$) $\mathcal{A} = \{a_0, a_1\}$ where a_0 represents maintainance without replacing the machine and a_1 denotes replacing the machine. For performing replace operation, a cost of 0.7 is required to be paid.

For the above MDP, the optimal policy turns out to be $[a_0, a_0, a_1, a_1]$. In the above policy, the thresholding nature is clearly visible. Once the agent shifts from action a_0 to action a_1, it never returns back to a_0 again.

If the structure is not known, then to find the optimal policy, we need to make a list of all the policies and their respective value functions. Once the list is made, the value functions are compared and the highest value function (or lowest value function in a cost-based setting) is declared to be the **optimal value function** and the policy corresponding to it is the **opitmal policy**.

However, if the threshold nature is known beforehand, then one can list the possible policies as $[a_0, a_0, a_0, a_0]$, $[a_0, a_0, a_0, a_1]$, $[a_0, a_0, a_1, a_1]$, $[a_0, a_1, a_1, a_1]$ and $[a_1, a_1, a_1, a_1]$. Comparing the value functions of the constructed policies by exploiting the structure is a linear time operation and can be completed in a linear time. This reduces the optimal policy search time significantly.

2 River swim problem

The river swim problem is a classic optimization problem in control theory and reinforcement learning. It involves a hypothetical situation where a swimmer needs to cross a river to reach the other side. The swimmer can choose to swim directly across the river or to swim diagonally upstream and then swim directly across the river.

The challenge is to find the optimal strategy that minimizes the swimmer's time to reach the other side. The swimmer's speed is greater when swimming directly across the river, but diagonal swimming consumes less energy due to the slower water current.

The river swim problem can be formulated as a Markov decision process (MDP), where the swimmer's actions are the choices between swimming directly across the river or diagonally upstream, and the state transitions are determined by the water current and the swimmer's speed. The reward is the negative of the time taken to reach the other side.

The solution to the river swim problem can be found by solving the MDP using dynamic programming or reinforcement learning methods. One of the most commonly used methods is Q-learning, which is a model-free reinforcement learning algorithm that learns an optimal policy from the observed rewards and transitions.

The river swim problem has applications in various domains, including robotics, where it can be used to optimize the path planning of autonomous vehicles moving in the water. It also has implications for decision-making in scenarios where a series of choices need to be made, and the optimal outcome is not immediately apparent.

However, here we solve a modified version of the classic River swim problem. The number of states is six and the number of actions is two. This benchmark represents a situation when a swimmer has to swim across a river in order to reach one of the coasts.

In between the two coasts, there are four islands. The swimmer is present at any of the states (either islands or one coast). The river current is from the right to the left side. At each state, the swimmer can either stay in the same state, swim left (in the direction of the river current) or swim right (in the direction opposite to the river current). The reward value is greater than zero at both the coasts (first and last state). In all other states, the reward value is zero. So the aim of the agent is to transition to the nearest coast (if the rewards were symmetric).

In this paper, the rewards are not symmetric(as shown in figure 1). This is done in order to verify whether complex problems and decision-making are handled by the algorithms mentioned in this paper or not. For this benchmark, the algorithms in the paper are tested and compared with two variants of Q-learning (as mentioned under the Machine Replacement problem). The transition probability function and reward function is shown in the table 1 The transition state diagram for the River swim problem solved by us is as shown in figure 2.

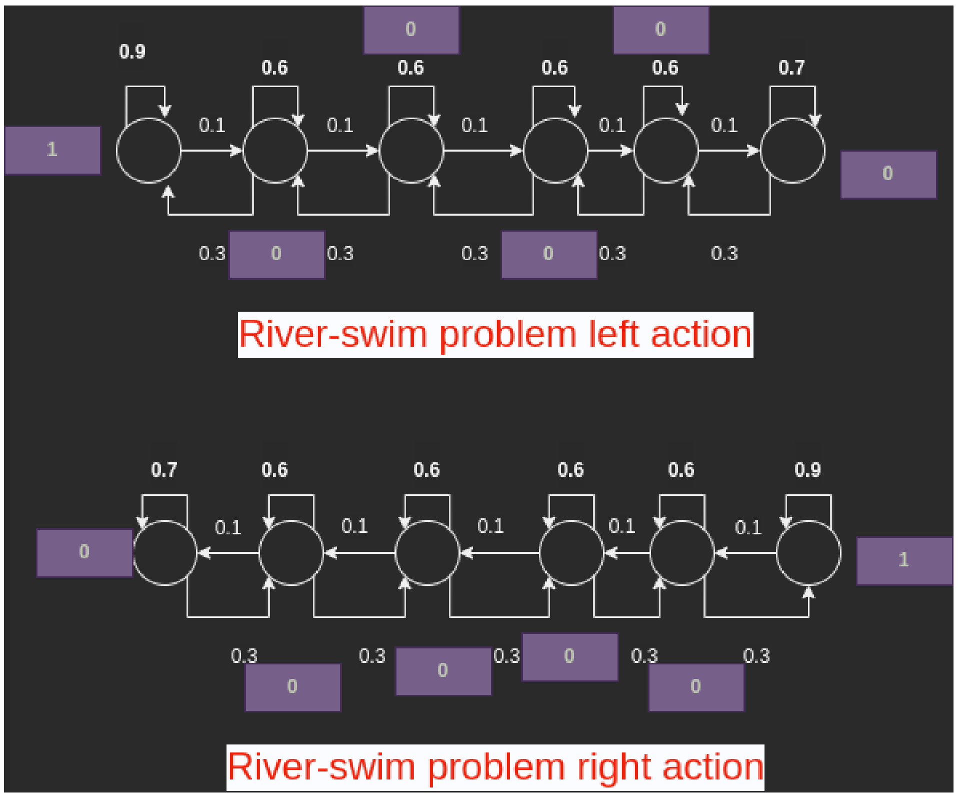

Figure 1: River swim state transition diagram if the model is symmetric about the two actions

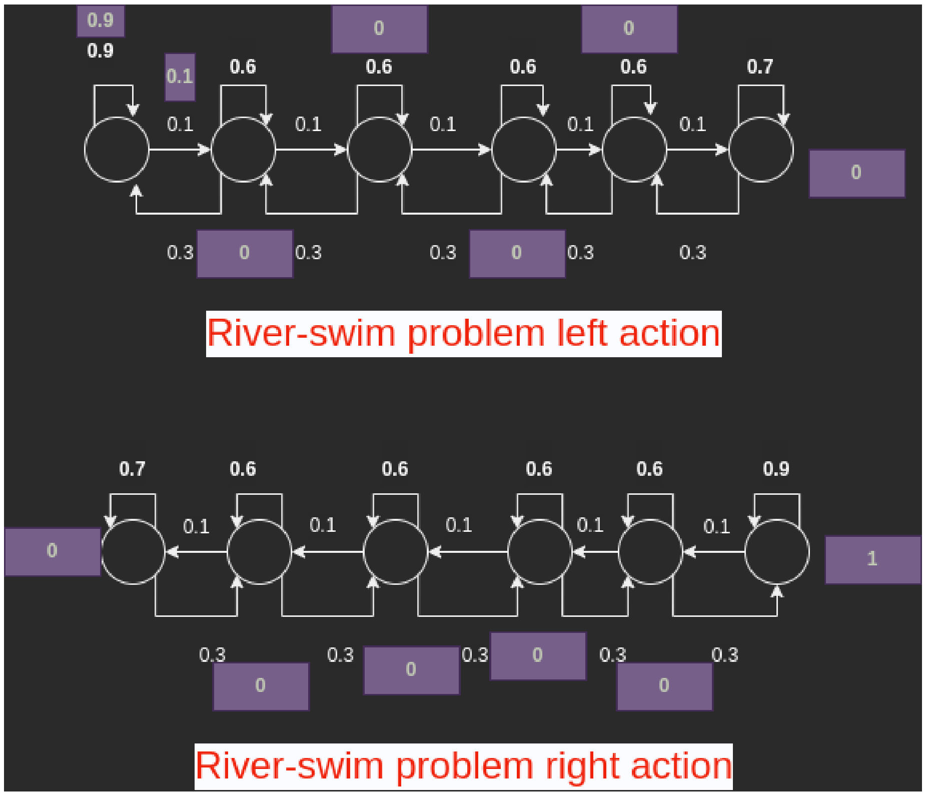

Figure 2: River swim state transition diagram if the model is not symmetric about the two actions

state	next state	action	transition probability	reward
6	6	R	0.9	1
6	5	R	0.1	1
s	s+1	R	0.3	0
s	s	R	0.6	0
s	s-1	R	0.1	0
0	0	L	0.9	0.9
0	1	L	0.1	0.9
s	s-1	L	0.3	0
s	s	L	0.6	0
s	s+1	L	0.1	0

Table 1: Transition Probability and reward function. R in the action column stands for swim-Right action and L in the action column stands for swim-Left action. For R action state 6 has different transition probability and reward functions so mentioned separately. Also for L action, state 0 has separate transition probability and reward value so mentioned separately