

IT 567 - Lab Exercise 1

Winter 2024-2025

Kewalramani, Tanay
202201362

14 January 2025

Problem 1: Recycling Robot

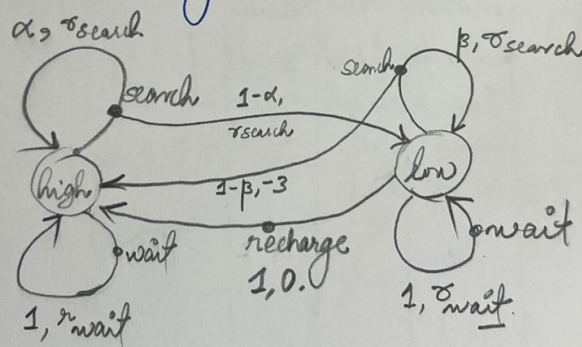
The solution to Problem 1 is below. The solution is scanned and attached.
(see next page)

Lab-1 (January 13th 2025)

Problem 1 [Example 3.3]

Recharging robot.

The robot has 2 guaranteed states - high, low.



each pair (a, b) over a transition denotes
 $a \rightarrow$ probability
 $b \rightarrow$ reward.

Figure 1: First part of solution for problem 1.

$states \Rightarrow \{high, low\}$
 $actions \Rightarrow \{search, wait, recharge\}$
 so, some s, a, s' states
 won't make sense (like
 $low \rightarrow recharge \rightarrow low$)
 We'll have $2 \times 3 \times 2$
 $= 12$
 entries in the table
 (s, a, s') ~~not~~ representing
 $3 \times 4 \times 5$ cartesian
product.

Figure 2: Second part of solution for problem 1.

Python Simulation

The Python simulation for the Recycling Robot is provided below. The code simulates the model for 10 time steps.

Code:

```

1 import random
2
3 states = ["high", "low"]
4 actions_high = ["search", "wait"]
5 actions_low = ["search", "wait", "recharge"]
6
7 alpha = 0.9
8 beta = 0.1
9 r_search = 1
10 r_wait = 0.5
11

```

```

12 transition_table = {
13     "high": {
14         "search": [("high", alpha, r_search), ("low", 1
15             - alpha, r_search)],
16         "wait": [("high", 1.0, r_wait)],
17     },
18     "low": {
19         "search": [("low", beta, r_search), ("high", 1 -
20             beta, -3)],
21         "wait": [("low", 1.0, r_wait)],
22         "recharge": [("high", 1.0, 0)],
23     },
24 }
25
26 def simulate_step(current_state):
27     if current_state == "high":
28         action = random.choice(actions_high)
29     elif current_state == "low":
30         action = random.choice(actions_low)
31     transitions = transition_table[current_state][action
32         ]
33     next_state = random.choices(
34         [t[0] for t in transitions], weights=[t[1] for t
35             in transitions], k=1
36     )[0]
37     reward = next(t[2] for t in transitions if t[0] ==
38         next_state)
39     return current_state, action, reward, next_state
40
41 current_state = "high"
42 print(f"{'Time':<5}_{'State':<10}_{'Action':<10}_{'
43     Reward':<10}_{'Next_State':<10}")
44 for t in range(1, 11):
45     state, action, reward, next_state = simulate_step(
46         current_state)
47     print(f"{t:<5}_{'state':<10}_{'action':<10}_{'reward':<10}
48         _{'next_state':<10}")
49     current_state = next_state

```

Output:

Time	State	Action	Reward	Next State
1	high	wait	0.5	high
2	high	search	1	high

3	high	wait	0.5	high
4	high	wait	0.5	high
5	high	search	1	high
6	high	search	1	low
7	low	search	-3	high
8	high	wait	0.5	high
9	high	wait	0.5	high
10	high	wait	0.5	high

Problem 2: Exercise 3.4

The solution to Problem 2 is below. I have attached my solution in the image.

Problem 2 We need to give a table w/ $p(s', r | s, a)$ including $s, a, s', r, p(s', r | s, a)$

s	a	s'	r	$p(s', r s, a)$
high	search	high	r_{search}	α
high	search	low	r_{search}	$1 - \alpha$
low	search	low	r_{search}	β
low	search	high	-3	$1 - \beta$
low	wait	low	r_{wait}	1
low high	wait	high	r_{wait}	1

only entries where reward $\neq 0$ is required. This is almost the same as the table in the example.

Figure 3: Handwritten solution for Problem 2.