IE 370, Spring 2024 Homework 6

Chris Lee

March 31, 2024

[ ]:
```python
from fractions import Fraction
from math import exp

from scipy.optimize import fsolve
from scipy.stats import poisson
```

1.1 (15 points) What is the probability that the number of forward jumps it takes her to jump over 9 for the first time (in forward direction), is exactly 4?

[ ]:
```python
rate = 1/2
t = 9

# The number of events (jumps) before going over t, since we are looking for␣
↪the 4th to be beyond 9
n = 3

prob = poisson.pmf(n, rate*t)
prob
```

[ ]: 0.168717884924555

1.2 (15 points) What is the probability that the number of backward jumps N it takes her to jump over 9 for the first time (in backward direction), is exactly 4?

- First Jump Not Over 9: The probability that the first backward jump does not overshoot 9 is calculated using the memoryless property of exponential distributions. Given that the mean of the backward jump is 1 and the overshoot has a mean of 2, we find

$$(P\{Y < Z\} = \frac{2}{3})$$

- Memoryless Property: Due to the memoryless property, the distribution of the distance after the first jump, assuming it did not overshoot, remains exponential with the same mean as the initial overshoot distance.

- Recursive Probability: The overall probability ($P\{N = 4\}$) is determined by the recursive relationship involving the probabilities of needing fewer jumps, ultimately reducing to ($P\{N = 1\}$).

$(P\{N=1\})$ is $(1-P\{N>1\})$, which is $(\frac{1}{3})$ because $(P\{N>1\}=\frac{2}{3})$. $(P\{N=4\})$ equals $((\frac{2}{3})^3 \times \frac{1}{3})$, considering three instances where the jump is shorter than the overshoot and the final instance aligning with the required condition.

[ ]:
```
prob_N_equals_4 = (2/3)**3 * (1/3)
prob_N_equals_4
```

[ ]: 0.0987654320987654

Problem 2: What is the probability that Bob will take a bus before Alice?

[ ]:
```
rate_R1 = 1/20
rate_R2 = 1/10
time_interval = 30

prob_no_R1_buses = exp(-rate_R1 * time_interval)

prob_first_bus_R2 = rate_R2 / (rate_R1 + rate_R2)

prob_Bob_before_Alice = prob_no_R1_buses * prob_first_bus_R2

prob_Bob_before_Alice
```

[ ]: 0.1487534400989532

Problem 3. Can this process be modeled as a CTMC? If so, what is the state space and transition rates (the G_{ij} 's)? Yes, process can be modeled as a CTMC

State Space:

1. **A1**: Alice holds the book and clock H1 is set.
2. **A2**: Alice holds the book and clock H2 is set.
3. **B**: Bob holds the book.
4. **C**: Chris holds the book.

Transition Rates (G_{ij})**:**

From **Alice (A1)**, when H1 rings: - To Chris (**C**) with probability 1/4, $= 4 * 1/4 = 1$ (since $EXP(4)$ with 1/4 chance). - To herself but setting H2 (**A2**) with probability $3/4 = 4 * 3/4 = 3$.

From **Alice (A2)**, when H2 rings: - To Bob (**B**) with probability $1/2 = 4 * 1/2 = 2$ (since $EXP(4)$ with 1/2 chance). - To Chris (**C**) with probability $1/2 = 4 * 1/2 = 2$.

From **Bob (B)**: - To Alice (**A1**) = 2 (alarm B-A, $EXP(2)$). - To Chris (**C**) = 5 (alarm B-C, $EXP(5)$).

From **Chris (C)**: - To Alice (**A1**) = 5 (alarm C-A, $EXP(5)$). - To Bob (**B**) = 3 (alarm C-B, $EXP(3)$).

Problem 3. In the long-run, what is the fraction of time that Alice holds the book?

[ ]:
```python
def equations(vars):
    pi_B, pi_C, pi_A1, pi_A2 = vars
    eq1 = pi_C * 8 - (pi_B * 5 + pi_A1 * 1 + pi_A2 * 2)
    eq2 = pi_B * 7 - (pi_C * 3 + pi_A2 * 2)
    eq3 = pi_A2 * 4 - pi_A1 * 3
    eq4 = pi_B + pi_C + pi_A1 + pi_A2 - 1
    return [eq1, eq2, eq3, eq4]


initial_guesses = [0.25, 0.25, 0.25, 0.25]
pi_B, pi_C, pi_A1, pi_A2 = fsolve(equations, initial_guesses)
fraction_alice_holds_book = pi_A1 + pi_A2

display(
    f"In the long-run, the fraction of time that Alice holds the book is␣
 ↪{Fraction(fraction_alice_holds_book).limit_denominator()},␣
 ↪{fraction_alice_holds_book}"
)
```

```
'In the long-run, the fraction of time that Alice holds the book is 287/465, 0.
 ↪6172043010752688'
```

Problem 4. In the long-run, what is the average rate at which calls are actually taken for service?

[ ]:
```python
lambda_arrival = 1
mu_J = 1  # John's service rate (calls per hour)
mu_M = 3 / 2  # Mary's service rate (calls per hour)


def balance_equations(vars):
    pi_0, pi_J, pi_M, pi_2 = vars
    eq1 = pi_0 - (pi_J * mu_J + pi_M * mu_M)
    eq2 = pi_J * mu_J - pi_2 * mu_M
    eq3 = pi_M * mu_M - pi_2 * mu_J
    eq4 = pi_0 + pi_J + pi_M + pi_2 - 1
    return [eq1, eq2, eq3, eq4]


initial_guesses = [0.25, 0.25, 0.25, 0.25]
pi_0, pi_J, pi_M, pi_2 = fsolve(balance_equations, initial_guesses)
lambda_pi_0 = pi_0

display(
    f"In the long-run, the average rate at which calls are actually taken for␣
 ↪service is {lambda_pi_0} calls/hour"
)
```

```
'In the long-run, the average rate at which calls are actually taken for service␣
↪is 0.4411764705882353 calls/hour'
```