Names: Solutions

SA402 – Dynamic and Stochastic Models Fall 2013
Asst. Prof. Nelson Uhan

Lesson 23. Lab – Helicopter Maintenance

Instructions. You have the entire class period to complete this lab. You must work in teams of 3 or 4. Submit only 1 completed lab per team.

You have been asked to take over the task of staffing maintenance teams at a facility that repairs and maintains SH-60 Seahawk helicopters. The helicopters arrive at the maintenance facility at a rate of 16 per day, and are processed on a first-come-first-served basis. It takes a maintenance team 4 hours on average to process 1 helicopter. The facility currently has 3 teams working simultaneously at any given time. Your predecessor left you some MATLAB code that simulates the arrival and repair process as an M/M/3 queue.

1. First, get acquainted with what the code does. Download the code from the course website. Read through `experiment.m`, `simulation.m`, and `event.m`.

   a. What does `event(1)` do? What about `event(4)`?

   event(1) is the subroutine corresponding to a helicopter (customer) being served at "maintenance team 1."

   event(4) is the subroutine corresponding to a helicopter arriving at the facility.

   b. What do these lines in `simulation.m` do?

```
25  % Simulation stopping time
26  T_stop = 365;
```

```
78    % Stopping condition
79    if (T > T_stop)
80        break
81    end
```

These lines stop the simulation after T_stop days, which is set to 365 in the lines above.

2. What is the expected delay per helicopter? Use the formulas established in previous lessons.

$\lambda = 16$ helicopters/day
$\mu = 6$ helicopters/day
$s = 3$

$\Rightarrow \rho = \frac{\lambda}{s\mu} = \frac{8}{9}$ $\quad sp = \frac{\lambda}{\mu} = \frac{8}{3}$

$$\Rightarrow \pi_0 = \left[\sum_{j=0}^{3} \frac{(8/3)^j}{j!} + \frac{3^3 (8/9)^4}{3!\,(1/9)} \right]^{-1} = \frac{3}{107}$$

$$\pi_s = \frac{(8/3)^3}{3!}\pi_0 \approx 0.0886$$

$$\ell_q = \frac{8/9}{(1/9)^2}\pi_s \approx 6.3801 \text{ helicopters}$$

$$w_q = \frac{\ell_q}{\lambda} \approx 0.3988 \text{ days}$$

3. In `experiment.m`, write code that estimates the expected delay per helicopter via simulation. Do this by following the steps below – write your code snippets here.

   a. First, right before the main `for` loop, create an empty vector `delay` that will eventually hold the delay per helicopter in each generated sample path:

   ```
   delay = [];
   ```

   b. Next, in the main `for` loop, right after the simulation is run, make things simpler and create `R_path`, the sample path of the number of helicopters in the system:

   ```
   R_path = S_path(4,:)
   ```

   c. Use `R_path` to compute `Q_path`, the sample path of the number of helicopters in the queue.
   Hint. This can be done in 1 line in MATLAB.

   ```
   Q_path = max(R_path - s, 0)
   ```

   This also works:

   ```
   Q_path = R_path - sum(S_path(1:3,:))
   ```

d. Use `T_path` and `Q_path` to compute `total_delay`, the total delay of all helicopters for a given sample path. (No need for fancy MATLAB constructs – a simple `for` loop will do.)

```
total_delay = 0;
for i = 1 : length(Q_path) - 1
    total_delay = total_delay + (T_path(i+1) - T_path(i)) * Q_path(i);
end
```

e. Use `S_path` to compute `total_arrivals`, the total number of helicopter arrivals in a given sample path. (A simple `for` loop will suffice here as well.)

```
total_arrivals = 0;
for i = 1 : length(R_path) - 1
    if (R_path(i+1) == R_path(i) + 1)
        total_arrivals = total_arrivals + 1;
    end
end
```

f. Compute the delay per helicopter for a given sample path and add it to the vector `delay`:

`delay = [delay, <what goes here?>]` ← total_delay / total_arrivals

g. Right after the main `for` loop, compute `expected_delay`, the average of all the delays per helicopter from each sample path. This is an approximation of the expected delay per helicopter. What value do you obtain? It should be relatively close to your answer to question 2.

```
expected_delay = mean(delay)
```

Answers may vary – you should get something close to 0.3987.

h. Why might the value you obtained through simulation differ from the value you obtained analytically in question 2?

- We might not have used enough simulation replications.

- We might not be running each simulation long enough (i.e. Tstop needs to be larger) – remember that the performance measures computed in question 2 are long-run steady-state performance measures.

4. After analyzing the data, you determine that the service times are better modeled as a uniform random variable on [1/24, 7/24] (in days). Use Whitt's approximation with your answer to question 2 to approximate the expected delay. Note that the expected service time is still the same.

Interarrival times are the same $\Rightarrow \lambda = 16$, $\varepsilon_a = 1$

Service times $X \sim \text{Unif}\left[\frac{1}{24}, \frac{7}{24}\right] \Rightarrow \mu = 6$, $\varepsilon_s = \frac{\text{Var}(X)}{E[X]^2} = \frac{\frac{1}{12}\left(\frac{6}{24}\right)^2}{\left(\frac{1}{6}\right)^2} = \frac{3}{16}$

λ and μ are the same $\Rightarrow w_q \approx 0.3988$

$\Rightarrow \hat{w}_q = \left(\frac{\varepsilon_a + \varepsilon_s}{2}\right) w_q \approx \left(\frac{1 + \frac{3}{16}}{2}\right)(0.3988) \approx 0.2368$

5. Modify the simulation code to incorporate this new model of service times. Run the code to compute the expected delay over all generated sample paths. What value do you obtain? Is it close to your answer to question 4?

Change `exprnd(1/mu)` to `unifrnd(1/24, 7/24)` in event.m

$\Rightarrow$ Estimated delay is around 0.2340 (answers may vary)

The simulated value (the "real" value) is quite close to Whitt's approximation.

6. (Bonus) If the service times are exponential, how can you rewrite the simulation code to be more compact? Describe the changes that you would make. Why do your proposed changes work? *Hint.* You only need 3 events, instead of 5.

We don't need an event subroutine for each machine when the service times are exponential — we can simply have 1 event subroutine for customer/helicopter departures:

e_1():

```
S(4) = S(4) - 1
C(1) = T + exprnd(1/(min(S(4), s) * mu)
```

note: exprnd(1/0) = Inf, as desired when no customers/helicopters are being served.

Although it looks like we're resetting the time until the next departure every time a customer/helicopter departs, due to the memorylessness property of the exponential random variable, this still works.