DATA7202 Assignment 4

Longpeng Xu

June 9, 2023

Contents

1 Question 1

(a) **Problem summary**

Air Secure wishes to open some service desks. A discrete event system with simulations must be built to ensure the service efficiency, in terms of the waiting time of each customer and the number of service desks. According to its research, it is assumed that on arrival customer always choose the desk having the shortest queue. The inter-arrival time and service time of 1000 sampled customers are provided. Hence, this is a $M/M/n$ system, where customers keep arriving and there are multiple servers. If all the servers are busy, customers keep waiting until they are served. For this system, it is crucial that any convincing conclusions must be supported by steady-state samples exclusively. Moreover, in a simulation running over long-term, a batch-means method is used for deriving the results.

Project objective

The study aims to build a discrete event system, conduct simulations on it, and to decide the minimum number of services desks, ensuring that 90% of the customers wait no longer than 8 minutes before being served.

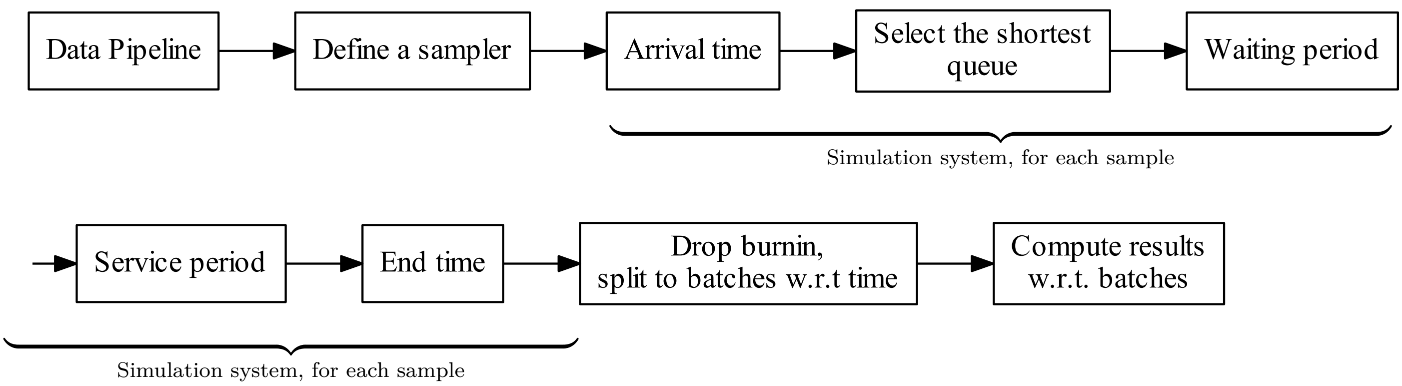
(b) **Parameters of the simulation function `DES()`**

`T`	Stopping time of the simulation, in minutes
`n`	Number of batches
`num_ser`	Number of service desks
`burnin`	The initial fraction as the burn-in period of `T`. The samples which arrived during the period should be thrown away

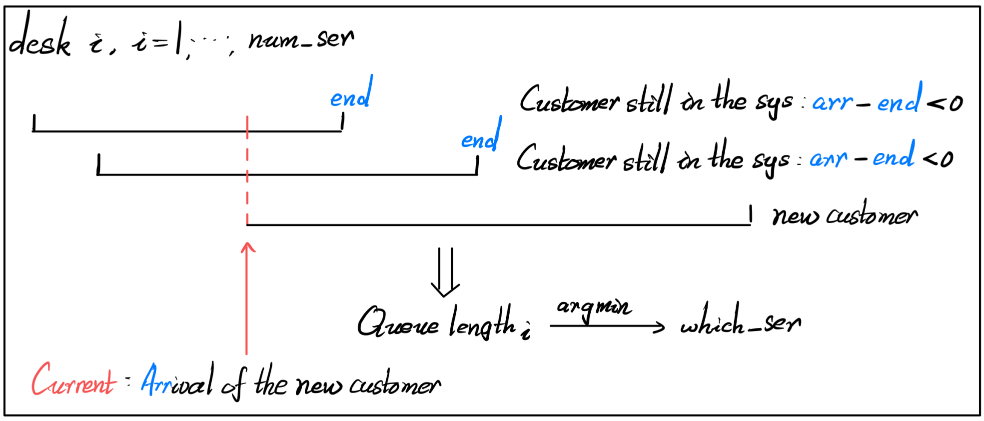
Other important variables within simulation

`arr_times`	List of each customer's `[arrival time, service desk id]`
`end_times`	List of each customer's `[end time, service desk id]`
`wait_periods`	List of each customer's waiting period
`pct_batches`	List of the percentage of the customers whose waiting period is no longer than 8 minutes, for each batch (i.e., the length of `pct_batches` = `n`)
`mean_batches`	List of the average waiting period of all the customers, for each batch (i.e., the length of `mean_batches` = `n`)
`arr_time`	The arrival time of a customer
`which_ser`	The service desk with the shortest queue at a customer's arrival
`wait_period`	The waiting period for a customer
`ser_period`	The service period for a customer
`end_time`	The leaving time of a customer
`batch_cutoff_time`	List of start and end times for each batches. Since the end time of a batch is the start time of the next batch, `batch_cutoff_time` is of length `n+1`
`batch_cutoff_ind`	List of start and end indices, in `wait_periods`, of each batches, which is mapped from `batch_cutoff_time`

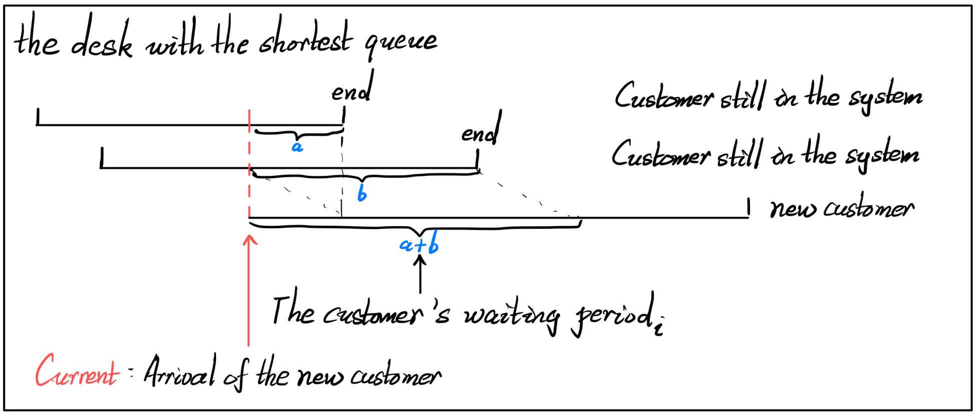
Project Dynamics



Select the shortest queue for a new customer (line 54-61 of code in Appendix)



Waiting period for a new customer (line 69-71 of code in Appendix)



(c) **Results**

(i) In a group of simulations where only the number of service desks is changed, the possible minimum number of service desks is 8.

(ii) In the simulations specified in (a), when there are 8 desks, The 95% CI of the percentage of the customers waiting no longer than 8 minutes is [0.983965, 0.996380], with mean 0.990173. The 95% CI of the waiting time of a customer is [0.363290, 0.640485] minute with mean 0.501887 minute. Figure 1 shows the $n = 50$ percentage figures versus different # desks. Figure 2 shows the histogram of the 50 percentage figures and that of the 50 waiting times.

Note: Using batch-means method, it may be assumed that the n batches are independent (Ed post #179). Hence, the CIs may be given by $\mu \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$ for large $n = 50$. If there are strict assumptions on batch independence, the CIs may be instead given by the $\alpha/2$, $1-\alpha/2$ percentiles of the observations, as in the tutorial session.

(iii) Despite that 8 is the possible minimum, in many groups of simulations, it is more likely that the minimum # desks required is 9. In another 20 groups of simulations, 12 groups shows that the minimum # is 9 while the other eight groups showing the minimum # is 8. Table 1 shows more information about the 20 groups of simulations.

(iv) According to Table 1, for those groups of simulations where the minimum # desks required is 8, having one more desk slightly increases the percentages (of customers waiting $\leqslant$ 8 min) and evidently decreases the waiting times.

(v) According to Table 1, in the 20 groups of simulations, there are 6 groups showing that if having 8 desks, all the customers wait more than 8 minutes and the waiting times are extremely large.

Figure 1 In a group of simulations, % of customers waiting $\leqslant$ 8 mins versus # desks

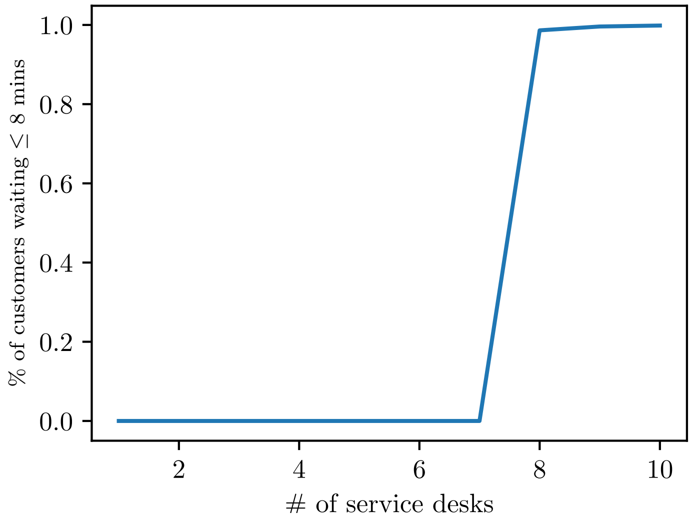


Figure 2 Left: Histogram of % customers waiting $\leqslant$ 8 mins; Right: Histogram of waiting times

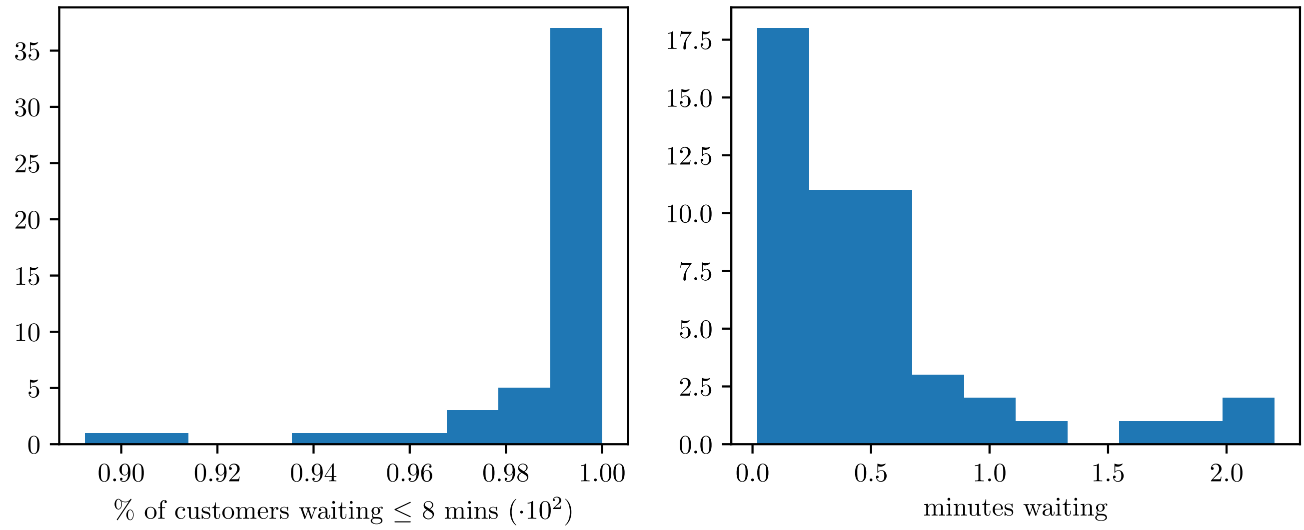


Table 1 % of customers wait $\leqslant$ 8 mins and mins waiting for 20 groups of simulations

Group	# desks	% customers wait $\leqslant$ 8 mins, average	mins waiting, average
1	8	0.00	inf
	9	99.59	0.18
2	8	98.62	0.55
	9	99.64	0.21
3	8	99.09	0.46
	9	99.59	0.20
4	8	48.15	inf
	9	99.68	0.20
5	8	99.11	0.47
	9	99.63	0.24
6	8	40.01	inf
	9	99.41	0.24
7	8	31.34	inf
	9	99.64	0.22
8	8	0.00	inf
	9	99.35	0.29
9	8	5.44	inf
	9	99.47	0.23
10	8	33.12	inf
	9	99.74	0.20
11	8	76.78	inf
	9	99.63	0.19
12	8	98.85	0.55
	9	99.70	0.19
13	8	27.05	inf
	9	99.50	0.24
14	8	91.79	1.43716E+11
	9	99.47	0.24
15	8	0.00	inf
	9	94.83	80500.89
16	8	98.86	0.55
	9	99.31	0.29
17	8	0.00	inf
	9	99.75	0.13
18	8	53.15	inf
	9	99.65	0.17
19	8	0.00	inf
	9	99.46	0.26
20	8	0.00	inf
	9	99.88	0.17

(d) **Conclusions**

(i) In Air Secure's $M/M/n$ system, by setting 9 desks can the company meet the 90%-within-8-minute requirement in a robust manner. If setting 8 desks, it is likely that no customers wait within 8 minutes but keep waiting infinitely before they are served.

(ii) Given the possible minimum # desks is 8, a cost-effective solution is to set 8 desks which are staffed and a flexible ninth desk. Setting a threshold for the total length of the queues, when the # queueing customers exceeds the threshold, a ninth staff member is allocated to the ninth desk.

(iii) The performance (e.g. % customers waiting $\leqslant$ 8 mins, mins waiting, etc.) is polarized when setting # desks to be the border value 8, which means that the steady state and "performance leap" of a system require carefully selecting parameters, beside dropping the burn-in samples.

(iv) Overall, the project objective of deciding minimum # of desks has been met, providing valuable decision-making support for Air Secure.

(e) See the fully-commented code in appendix. There is only one `.py` code file in the `.zip` package. Interaction: The code is in four parts: The first part is data pipeline. The second part builds a sampler, which is used in third part Simulation. The final part consists of test cases for the simulation.

2 Appendix

Listing 1: **Python Code**

```
# Data pipeline
import pandas as pd
interarr = pd.read_csv('data.csv', header=0, usecols=['inter_arrival_time'])
service = pd.read_csv('data.csv', header=0, usecols=['service_time'])



# Create sampler with replacement
import numpy as np

class sampler:
    def __init__(self, population):
        self.population = population
        self.remaining =  list(self.population)

    def sample(self, n):
                if n >  len(self.remaining):
            raise ValueError("Cannot sample more elements than remaining")
        sample = np.random.choice(self.remaining, size=n, replace=True)
        #self.remaining = [el for el in self.remaining if el not in sample]
        #print(f"The number of remaining samples is {len(self.remaining)}")
        return sample

# Create the instances
sampler_interarr = sampler(interarr.values.reshape(1,-1)[0])
sampler_service = sampler(service.values.reshape(1,-1)[0])



# Simulation
def DES(T, n, num_ser, burnin):
    '''
    T:       stop time
    n:  the number of batches
    num_ser: the number of servers
    burnin:  the first burnin% of the samples to be discarded
    '''
    arr_times = [[0.0000, 0]]
    end_times = []
    wait_periods = []
    pct_batches = []
    mean_batches = []
    step = 0

        # Continue simulation until T is reached
    while arr_times[-1][0] < T:

        # Compute the arrival time of a new customer via a sample from ECDF
```

```python
        if step == 0:
            arr_time = 0.0000
        else:
            arr_time = arr_times[-1][0] +  float(sampler_interarr.sample(1))

        # Enter into the shortest queue
        # queue_len is # customers still in the queue at the new customer's arrival
        # np.random.choice is used in case of multiple queues have the shortest length
        queue_len = np.array([])
        for i in  range(1, num_ser + 1):
            queue_len = np.append(queue_len,  len([j for j in end_times if j[1] == i and arr_time - j[0] < 0]))
        shortest =  list(np.argwhere(queue_len ==  min(queue_len)).reshape(1,-1)[0])
        which_ser = np.random.choice(shortest) + 1

        # Append the arrival time to arr_times
        if step == 0:
            arr_times[0][1] = which_ser
        else:
            arr_times += [[arr_time, which_ser]]

        # Compute the waiting period
        # the sum of the differences between arr_time (new customer) and end_time of each customers still in the shortest queue
        wait_period =  abs( sum([arr_time - k[0] for k in end_times if k[1] == which_ser and arr_time - k[0] < 0]))

        # Sample the service period
        ser_period =  float(sampler_service.sample(1))

        # Compute the end time
        end_time = arr_time + wait_period + ser_period
        end_times += [[end_time, which_ser]]

        # Append wait_period of current customer to wait_periods
        wait_periods += [wait_period]
        #print(f"Customer who chose server {which_ser}, with arrival time {round(arr_time,4)}, wait period {round(wait_period,4)}, service period {round(ser_period,4)}, end time {round(end_time,4)}.")
        step += 1

    # Compute the border index in arr_times for each batch, after burn-in
    batch_cutoff_time = np.linspace( int(T * burnin), T, n+1)
    batch_cutoff_ind = [ int(np.argwhere(np.array(arr_times)[:, 0] >= k)[0]) for k in batch_cutoff_time]

    # Compute the percentage of customers waiting < 8 min and the average time waiting for each batch
    for i in  range(n):
        start, end = batch_cutoff_ind[i], batch_cutoff_ind[i+1]
        pct_batches += [ round( len([m for m in wait_periods[start:end] if m<=8]) /  len(wait_periods[start:end]), 6)]
        mean_batches += [ round(np.mean(wait_periods[start:end]), 6)]

    #return pct_batches, np.mean(pct_batches), mean_batches, np.mean(mean_batches)
    return np.mean(pct_batches), np.mean(mean_batches)


```

```


# One group of simulations
for i in  range(5,11):
        mean_pct_batches, mean_mean_batches = DES(T=3000, n=50, num_ser=i, burnin=0.3)
        print(f"With {i} desks, {mean_pct_batches*100}% customers wait < 8 min (averaged across the batches), a customer waits for {mean_mean_batches} mins (averaged across the batches).")

# Another 20 groups of simulations
for j in  range(20):
  for i in  range(7,10):
    mean_pct_batches, mean_mean_batches = DES(T=3000, n=50, num_ser=i, burnin=0.3)
    print(f"With {i} desks, {mean_pct_batches*100}% customers wait < 8 min (averaged across the batches), a customer waits for {mean_mean_batches} mins (averaged across the batches).")
  print("=====================================================================")



# Plot flow chart for project dynamics
from graphviz import Digraph

dot = Digraph()
dot.attr(rankdir='LR') # horizontal arrangement

# Create nodes
dot.node('1', 'Data Pipeline', shape='box')
dot.node('2', 'Define a sampler', shape='box')
dot.node('3', 'Arrival time', shape='box')
dot.node('4', 'Select the shortest\nqueue', shape='box')
dot.node('5', 'Waiting period', shape='box')
dot.node('6', 'Service period', shape='box')
dot.node('7', 'End time', shape='box')
dot.node('8', 'Drop burnin,\nsplit to batches w.r.t time', shape='box')
dot.node('9', 'Compute results\nw.r.t. batches', shape='box')

# Create edges
dot.edges(['12', '23', '34', '45', '56', '67', '78', '89'])

# Save the flow chart
dot.render('7202a4fig0', view=True)

# Figure 1
import matplotlib.pyplot as plt
plt.rcParams.update({'text.usetex': True, 'font.family': 'serif'})
plt.figure(figsize=(4,3), dpi=600, facecolor='white')

x =  range(1,11)
y = (0,0,0,0,0,0,0, 0.990173, 0.996075, 0.998493)
plt.plot(x,y)
plt.xlabel('\# of service desks')
plt.ylabel("\% of customers waiting $\le$ 8 mins", fontsize=8)
plt.savefig('7202a4fig1.pdf')
```

```
plt.show()

# Figure 2
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(7, 3), tight_layout=True, dpi=600, facecolor='white')
axes[0].hist(pct_batches)
axes[0]. set(xlabel="\% of customers waiting $\le$ 8 mins ($\cdot 10^2$)")
axes[1].hist(mean_batches)
axes[1]. set(xlabel="minutes waiting")
plt.savefig('7202a4fig2.pdf')
plt.show()
```