SHANGHAI JIAO TONG UNIVERSITY

The analysis and simulation of queuing system in two fast food chain

Liong Khai Jiet
Advisor: Shi Jian Hong

A final project report submitted in partial fulfillment of the
course ICE-6502

August 2020

LIST OF CONTENTS

1 Introduction **1**

2 Material and methods **1**

2.1 Model . . . 1

2.1.1 Model of Customer Population . . . 2

2.1.2 Waiting Line Model . . . 2

2.1.3 Model of Service System . . . 3

2.1.4 Model of Kitchen Process . . . 4

2.2 Performance measure . . . 4

3 Results and Discussion **5**

3.1 System utilization rate . . . 5

3.2 The average number of customers waiting in line . . . 5

3.3 Average time of a customer spend in the system . . . 6

3.4 Average time of a customer spend waiting . . . 6

3.5 Average Service Times and its Distribution . . . 7

4 Conclusion **7**

5 Acknowledgment **8**

6 Appendix: Source code for simulation **i**

Abstract

Fast food restaurants are popular among working adults and students who value the conducive environment and its convenient services. As such, fast food chain like McDonald's (MCD) and Kentucky Fried Chicken(KFC) are available in most places including shopping complex, office area and university cafeteria in Malaysia. Fast-food restaurants illustrate the transient nature of waiting line system, as they introduce promotions value meal time to time, resulting occasional long queues and inconvenient waiting times. For fast food, as the names states means it has a short service time. Thus, it is a suitable target for this project to analyse the performance measure of the multiple server single queue system of a restaurant.

Keywords: blocking, erlang, two-stage queue

1. INTRODUCTION

Queuing theory is the mathematical study of waiting lines which are often used to make predictions about how a system could cope with demands [1]. In this project, we want to study the queuing system in two fast food chain affect the efficiency. The queue times or total waiting times is the most significant part of a good customer experience [2], so the simulation focuses on analyzing how the two type of queuing system affects the average waiting times of the customer and the idle time of the serving counters. Running a computer simulation is a efficient method to validate the analytical model of the system to pinpoint any bottleneck resources in the system. This also allows us to test the efficiency of solution approached by both fast food chain restaurant using stochastic and queuing theory.

The challenge here is to determine which service system has sufficient but not excessive amount of capacity which translates to cost in the business world. As for fast food restaurants, short queuing time and short waiting line is important in attracting more customers, putting aside the factors like good dining environment and delicious food [3]. In this report, I will focus on the two largest fast food chain in Malaysia [4] and observe their queuing pattern. MC Donald's is using the multi-server, two-phase ticketing queue where the customer order food from two counter and get a ticket while the cook prepare the food [5]. KFC use a multi-server, two-phase blocking queue where the cashier takes order from the customer, send it to the kitchen and then the payment is made when they receive the food. Although both are having single queue, they have different type of system which the former uses a number queue system for food collection and the latter uses a "blocking" style queuing system when customers transition to the next stage.

The objective of this project is to estimate how this blocking have impact upon the customer flow in a restaurant and its likelihood of a completely loaded system.

2. MATERIAL AND METHODS

2.1. Model

The queuing model designed for the computer simulation describes the probabilistic nature of the two different queuing system. It allows mathematicians to use the constricted model to make predictions and study how an inefficient queuing system could affect customer experience (long waiting lines) and resource wastage (low utilization rate). Most of the queuing problem could be classified into three parts, which are the input process, the service distribution and the queue discipline [6].

A unlimited queuing simulation $(t_1, t_2, \ldots, t_n)$ will be carried out using a python package Simpy [7] to simulate the restaurant operation hours. The following assumptions applied in the simulations are listed here and discuss in the corresponding sections:

a) Infinite calling population

b) First-come, first-served queue discipline

c) Poisson arrival rate, λ

d) Exponential service rate, μ_1, μ_2

e) Customers only buys one set of food

f) Customers does not return to the queue after completing a purchase.

g) Kitchen cooks can only prepare one set of food a time.

h) There is no waiting line between neighboring stages in the system with blocking.

The following notation is used to represent the random variables associated with the model:

$N_q =$ the number of customers in queue.
$N_s =$ the number of customers receiving service.
$N = N_q + N_s =$ the total number of customers in the system.

$X_s =$ the time of a customer spends in taking order.
$X_k =$ the time of a customer spends in waiting for food.
$X = X_s + X_k =$ the time of a customer spends in actual service.

$T = W + X =$ the total time a customer spend in the system.
$\lambda =$ the arrival rate (average number of arrivals/min)
$\mu =$ the service rate (average number served / min)
c = the number of counter services in parallel
s = the number of cooks in the kitchen
n = the number of customer served
S = System

In Kendall's notation,

2.1.1 Model of Customer Population

The simulation also assumed the customer population in the model used in this supplement are are infinite and patient, so they do not renege, balk or jockey to prevent the mathematical formula to become overly complex. Besides, the simulation will end when the the restaurant served 1000 customers (N_{1000}) the remaining customers will leave the system at once no matter which process they currently at. Therefore, the number of customer served will be only the ones who completed the whole process and the unfinished processes are discarded completely.

2.1.2 Waiting Line Model

In the experiment setting, the arrival rate of customers has a Poisson distribution with parameter λ, where each of them are allowed to only buy one set of food. Then, the customers will need to enter the waiting line with infinite N the customers can place order via **r** number of serving counters and the order is then sent to

the kitchen to be prepared by **s** number of cooks. Each counter server and kitchen cook can only handle one task meal at a time, i.e. placing order and preparing food. Besides, the process of the customer placing order and kitchen preparing the meal are independent and has the exponential distribution with parameter μ. The sequence of arrival process and placing order are all assumed to be mutually independent.

The system describes the characteristics of queue system, such as the number of waiting lines, the number of server and service patterns, etc. The two variables introduced in the experiment is the queue with blocking or without blocking, before the transition to the second stage of service point (food collection) which typically arises from the problem of lacking queuing capacity between jobs [8].

Both model could be broken down into two stages where the first stage are identical in the sense that they have single-line, multiple server with First-come-first-serve discipline (M/M/c). The difference is multi-stage tandem-queuing model [9] system showed in figure 1 and the latter multi-stage wait-queuing model (G/M/∞) [10] showed in figure 2

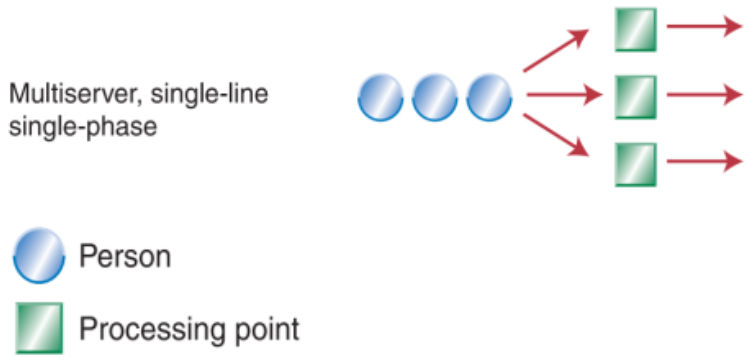

Fig. 1. Single-phase line model (w/ blocking)

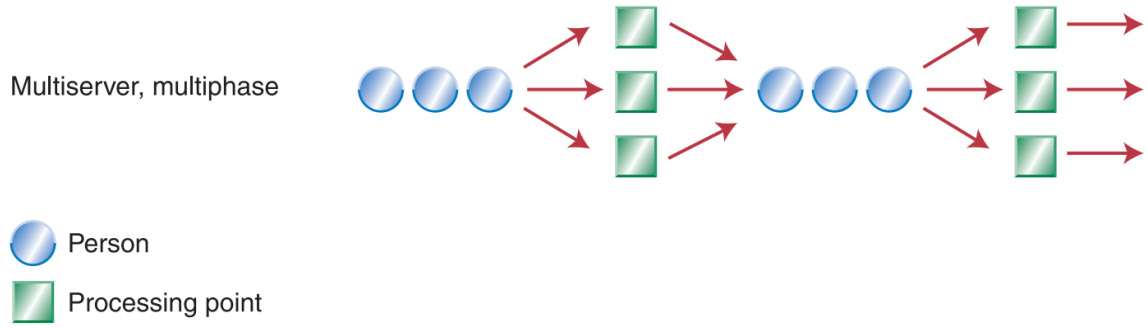

Fig. 2. Multi-phase line model (w/o blocking)

2.1.3 Model of Service System

In this project, we will study two types of queuing method, which are as follows:

i) S_1: Single line, Multi-server, Two-stage (w/ blocking) (Figure 1)
Scenario: Customer arrival rate follows the Poisson process with the parameter λ. The arrived (n)th customer has to pass through two consecutive service point before leaving the system. The (n)th customer will enter the only one waiting line with unlimited queue space at the first stage of serving point. Then, the customer will enter one of the free **r** counters and place an order but because there is no waiting space at the second stage of serving point. So the (n)th customer will need to wait at the counter while the kitchen is preparing the food.

The kitchen will process the resources with first-come, first-served priority rule. The server takes order from the customer with the service rate of exponential distribution parameter μ_1.

Next, the order is sent to the kitchen and the customer exits the system after they make the payment and receive the food.

ii) S_2: Single line, Multi-server, Two-stage (w/o blocking) (Figure 2)
Scenario: Customer arrival rate follows the Poisson process with the parameter λ; The arrived (n)th customer has to pass through two consecutive service point before leaving the system. The customers will enter the only one waiting line with unlimited queue space at the first stage of serving point. Then, the (n)th customer will enter one of the free **r** counters and place an order but because there is no waiting space at the second stage of serving point. So the customer will be blocked at the counter upon finish placing the order while the kitchen is preparing the food.

The kitchen will process the resources with the first-come,first-served priority rule. The server takes order from the customer with the service rate of exponential distribution parameter μ_1.

After making the payment, the counter will send the order to the kitchen and starts to service the next customer whereas the customer proceeds to the collection area. The customer exits the system after they receive the food.

2.1.4 Model of Kitchen Process

Cooks are the main resources in the kitchen. We model the ***s*** cooks as fixed resources. Once an item order (i)th arrives, it need to enter the order waiting queue, β_k for a cook to be free. Once the order is assigned to the cook, the cook will start preparing the food (i)th next in queue and cannot prepare $(i+1,\ i+2,$ ..., $i_n)$th orders simultaneously. Thus, the food preparation time is modeled using the exponential distribution of parameter μ_2. Once complete, the cook will send the (i)th order back to the counter and the process is repeated for $(i+1,\ i+2,$ ..., $i_n)$th order.

2.2. Performance measure

The project focus on few metrics to analyze the queuing system model and obtain the estimation of their corresponding performance measure. Before that, we create the following project scope and performance metrics to help us understand and identify any bottleneck found in the two systems.

From the section above, we introduced that

$$\lambda = \text{ mean arrival rate}$$
$$\mu = \text{mean service rate}$$

Now, we use Little's Law to derive the formula below:

$$\rho = \frac{\lambda}{c\mu} = \text{System utilization rate.}$$
$$L_Q = \rho\dot{L} = \text{ Average number of customers waiting in line and its distribution.}$$
$$W = \frac{1}{\mu - \lambda} = \text{Average time of a customer spends in the system.}$$
$$W_Q = \text{ Average time of a customer spend waiting.}$$

The data findings allows us to understand the relationship between the number queuing method, average waiting time and system utilization percentage.

3. RESULTS AND DISCUSSION

In the simulation, the system with stage blocking are assigned to the name **FIFO** whereas the system without stage blocking are assigned with the name **TICKET**.

3.1. System utilization rate

With $\lambda = 3$, $\mu = 1.5$ we plot the graph of how the number of servers affect the efficiency of each server. We can observe that **FIFO** system has lower utilization percentage (high idle times) because of the blocking mechanism at the second stage. The server resource is idle but it is blocked therefore it cannot process take order from another customer. In business context, this idling duration translates to business cost. Thus, the **TICKET** system with a waiting line in between the two stages can eventually maximize the resource utilization hence reaching the goal of 1000 customers served about 40% faster compared to **FIFO**.

The results are tabulated in table 1 with the parameters, $\lambda = 2, \mu_1 = 2$ and $\mu_2 = 2$

Table 1: Simulations results with n = 1000

	FIFO	**TICKET**
time unit	minute	
Total Simulation Time	909	513
Average Queue Length	223	5
Max Queue Length (min)	450	27
Average Waiting Time (min)	205	4.65
Average Service Time (min)	4.5	4.51
Average Time Spent in System (min)	207.33	8.51

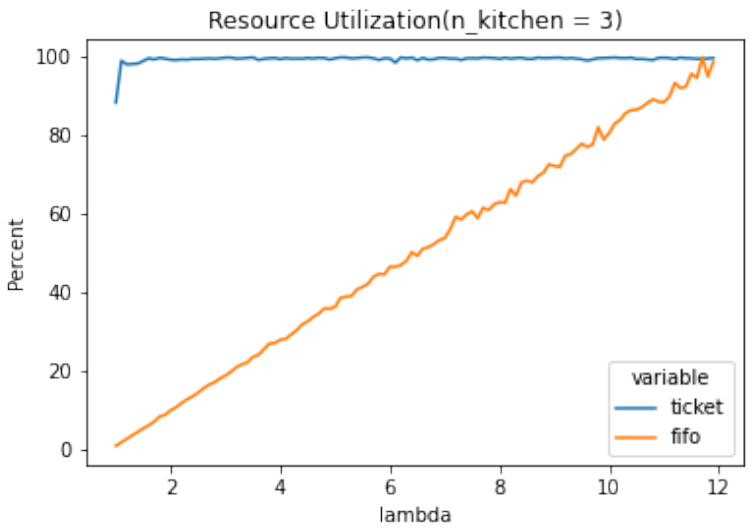

Fig. 3. ρ = Relationship between λ and system utilization percentage

3.2. The average number of customers waiting in line

The simulation results in left diagram in 4 shows that average queue length of **FIFO** has the higher exponential rate of increase when arrival rate λ increases compared to the **TICKET** system. The right graph also showed similar results when the service rate μ decreases, customers in **FIFO** system will have higher probability of facing longer queue.

Indeed, the **FIFO** system with blocking is observed to be more easily affected by the increase of arrival rate (i.e. lunch hour) or decrease of service rate (staff shortage).

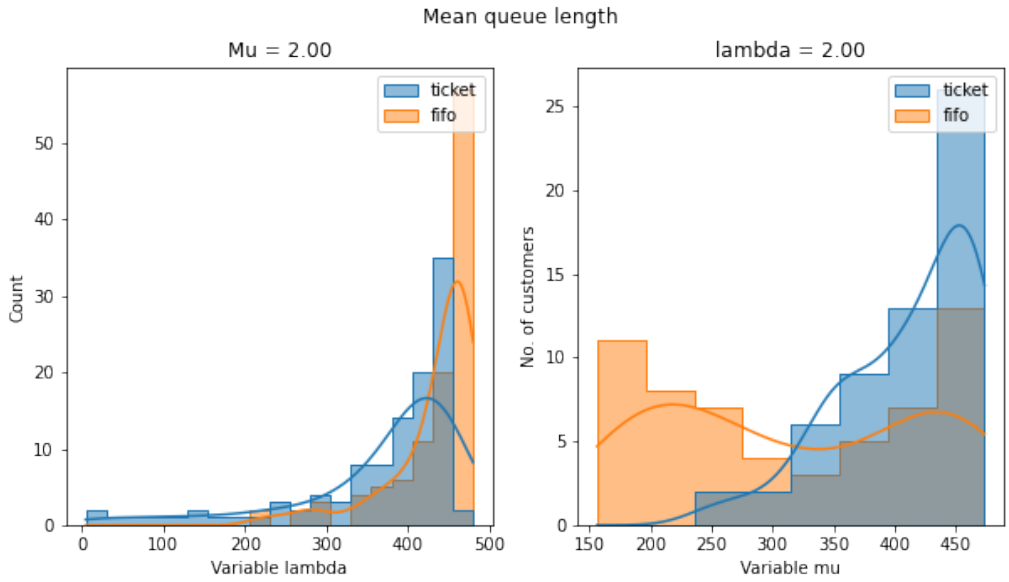

Fig. 4. L_Q = Average queue length

3.3. Average time of a customer spend in the system

The average time of a customer spend in the system includes both queuing and service time. The simulation results shows similar conclusion where customers in **FIFO** system has a higher probability of spending more time in the system.

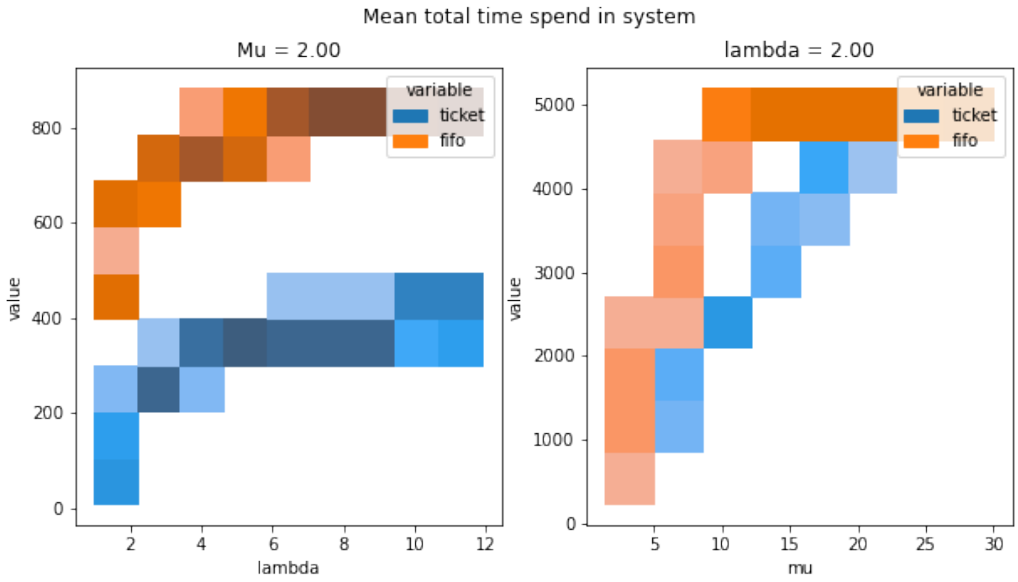

Fig. 5. W_Q = Average time in system

3.4. Average time of a customer spend waiting

The average time of a customer spend waiting includes both queuing and food collection process. The simulation results shows similar conclusion where customers in **FIFO** system has a higher probability of spending more time waiting which is also observed in the simulation results compiled in table 1.

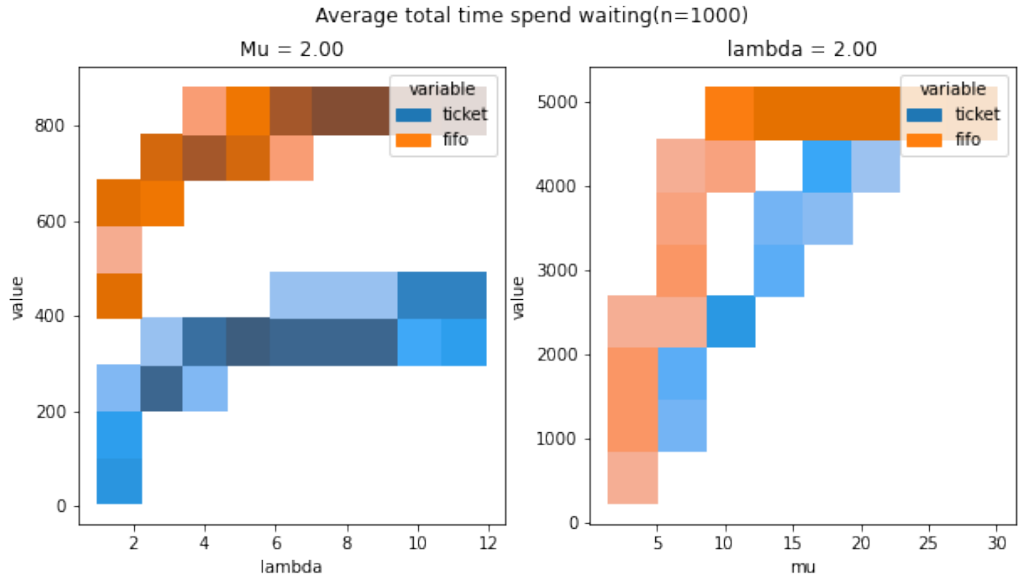

Fig. 6. W_Q = Average time spend waiting

3.5. Average Service Times and its Distribution

As we discussed earlier, the model has two service rate, μ_1 and μ_2 which both follows the exponential distribution as shown in the left graph 7. Indeed, when the service times from two stages are combined, the simulation result shows an Erlang distribution as proved in previous studies [11, 12]

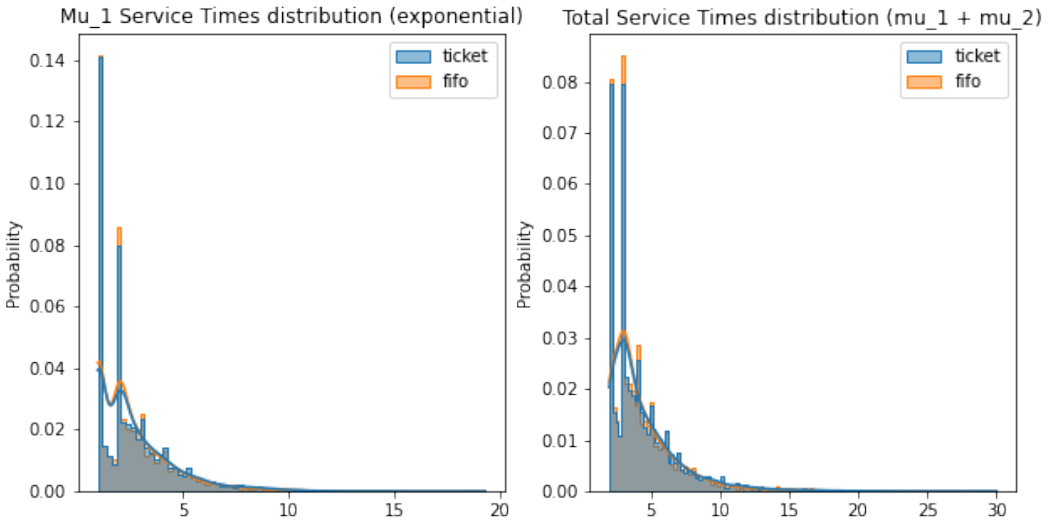

Fig. 7. = Service Times and its distribution

4. CONCLUSION

In this study, we simulate the blocking problem in fast food chain and analyze the impacts of the service interaction and customer experience. We showed that the multi-tandem stage system has a correlation between the blocking and waiting times at the following stages for later customers. Based on personal experience and reviews from Google, the queue length and expected queue time increases significantly during lunch hours or

when the restaurants offer limited merchandise in happy meals, long queue is more common in KFC restaurants because of the efficiency of queuing system used compared to MCD. Although in practice, KFC servers most probably will ask the customer to wait aside while taking order for the next customer, simulation results still proved that MCD wait queuing system is more efficient, reduced wait times, have better staff utilization, customer flow management and minimizing operational cost.

5. ACKNOWLEDGMENT

I am eternally grateful to my advisor, Prof. Shi Jian Hong for explaining about stochastic processes and queuing theory which provided me a good understanding and knowledge to complete this project. I would also like to thank my friend for her willingness to proof read and provide useful suggestions for my work. Finally, I would like to thank SJTU for allowing us to proceed our studies online in times of hardship like this and the canvas technical teams for the great support during our semester.

Last but not least, I also gain much knowledge and experience on how to model real life problems into computer simulation which is very useful for my later studies.

REFERENCES

[1] Adan, I. and Resing, J. "Queueing theory". In: (2002).

[2] Roy, D., Bandyopadhyay, A., and Banerjee, P. "A nested semi-open queuing network model for analyzing dine-in restaurant performance". In: *Computers & Operations Research* 65 (2016), pp. 29–41. ISSN: 0305-0548. DOI: 10.1016/j.cor.2015.06.006. URL: http://www.sciencedirect.com/science/article/pii/S0305054815001513.

[3] Dharmawirya, M., Oktadiana, H., and Adi, E. "Analysis of expected and actual waiting time in fast food restaurants". In: *Industial Engineering Letters* 2.5 (2012).

[4] Abdullah, N. N. et al. "Trend on fast food consumption in relation to obesity among Selangor urban community". In: *Procedia-Social and Behavioral Sciences* 202 (2015), pp. 505–513.

[5] Koh, H. L. et al. "Improving queuing service at McDonald's". In: *AIP Conference Proceedings* 1605.1 (2014), pp. 1073–1078. DOI: 10.1063/1.4887740. URL: https://aip.scitation.org/doi/abs/10.1063/1.4887740.

[6] Pinsky, M. A. and Karlin, S. "9 - Queueing Systems". In: *An Introduction to Stochastic Modeling (Fourth Edition)*. Ed. by M. A. Pinsky and S. Karlin. Fourth Edition. Boston: Academic Press, 2011, pp. 447–494. ISBN: 978-0-12-381416-6. DOI: 10.1016/B978-0-12-381416-6.00009-5. URL: http://www.sciencedirect.com/science/article/pii/B9780123814166000095.

[7] Grayson, P. *team-simpy / simpy*. URL: https://gitlab.com/team-simpy/simpy.

[8] Gómez-Corral, A. "A Tandem Queue with Blocking and Markovian Arrival Process". In: *Queueing Systems* 41.4 (2002), pp. 343–370. ISSN: 1572-9443. DOI: 10.1023/A:1016235415066.

[9] Ross, S. M. *Introduction to probability models*. Academic press, 2014.

[10] Wu, X., Li, J., and Chu, C.-H. "Modeling multi-stage healthcare systems with service interactions under blocking for bed allocation". In: *European Journal of Operational Research* 278.3 (2019), pp. 927–941. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2019.05.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221719303881.

[11] Langaris, C. "The Waiting-Time Process of a Queueing System with Gamma-Type Input and Blocking". In: *Journal of Applied Probability* 23.1 (1986), pp. 166–174. ISSN: 00219002. URL: http://www.jstor.org/stable/3214125.

[12] Ausin, M. C. "Bayesian estimation for the M/G/1 queue using a phase-type approximation". In: *Journal of Statistical Planning and Inference* 118.1 (2004), pp. 83–101. ISSN: 0378-3758. DOI: 10.1016/S0378-3758(02)00398-1. URL: http://www.sciencedirect.com/science/article/pii/S0378375802003981.

6. APPENDIX: SOURCE CODE FOR SIMULATION

```
import simpy
import numpy as np
import pandas as pd
from numpy import random
import math
from helper import round_down

counter_total_service_times = 0
counter_total_idle_times = 0
column_names = ["Arrival Time","Current Q length","Q time","Service Start Time","Food prepare start","Exit system time","Food Prepare Duration","Total Wait Time(Queue+Food)","Service Time","Total Time in System"]

class Counter(object):
#     Counters to take order
    def __init__(self,env,num_counter):
        self.env = env
        self.counter = simpy.Resource(env,num_counter)
        self.counter_waiting = 0
        self.service_start = None

    def take_order(self,cus,env,service_start,parameters):
        print("%s is placing order at counter %.2f" %(cus,service_start))
        time_taken_to_place_order = max(random.exponential(scale = parameters['order_time_mu']),parameters['order_time_min'])
        yield self.env.timeout(time_taken_to_place_order)
        print("Order of %s sent to kitchen at %.2f" %(cus, env.now))
        # Record idle counter and add to total count


    def receive_order(self,cus,env,resource,service_start,parameters,data):
        global counter_total_idle_times
        global counter_total_service_times

        with resource.kitchen.request() as my_turn:
            yield my_turn
            yield env.process(resource.prepare_food(cus,env,data,parameters))
            service_end = env.now
            print("%s collected the food at %.2f" %(cus, service_end))
            if(cus<1000):
                counter_total_service_times += (service_end-service_start)


class Kitchen(object):
    # Kitchen to prepare food
    def __init__(self,env,num_kitchen):
        self.env = env
        self.kitchen = simpy.Resource(env,num_kitchen)

    def prepare_food(self,cus,env,data,parameters):
        print("Kitchen is preparing food for %s at %.2f" %(cus, env.now))
        food_prepare_time = max(parameters['food_prepare_min'],random.exponential(scale = parameters['food_prepare_mu']))
        data[cus,6] = round_down(food_prepare_time)
        yield self.env.timeout(food_prepare_time)
        print("Cooked food for %s at %.2f" %(cus, env.now))

```

```
def customer(env, label, queue, kitchen,parameters, data):
#     the customer process arrive at the restaurant and request counter to take order
    label = label-1
    arrive_time = env.now
    print("%s entering the queue at %.2f"%(label,arrive_time))
#     data[label,0]=label
    data[label,0]= round_down(arrive_time)
    data[label,1] = len(queue.counter.queue)
    with queue.counter.request() as my_turn:
        yield my_turn
        service_start = env.now
        data[label,3] = round_down(service_start)
        queue_time = service_start - arrive_time
        data[label,2]= round_down(queue_time)
        # placing order at counter
        yield env.process(queue.take_order(label,env,service_start,parameters))
        # waiting order at counter
        prepare_food_start = env.now
        data[label,4] = round_down(prepare_food_start)
        # counter is idle now
        yield env.process(queue.receive_order(label,env,kitchen,service_start,parameters,data))
        # prepare_food_end = round_down(env.now)
        # counter_total_wait_times += round_down(prepare_food_end - prepare_food_start)
        # receive food from counter
        exit_time = env.now
        data[label,5] = round_down(exit_time)

        # total wait time
        data[label,7] = round_down(data[label,6]+data[label,2])
        # total service time
        data[label,8] = round_down(exit_time-service_start)
        # total time in system
        data[label,9] = round_down(exit_time-arrive_time)
        yield env.timeout(0)


# Simlating possion process for customer arrival
def customer_arrivals(env,n_customer,res_counter,kitchen,parameters,result_fifo):
    """Create new *customer* until the sim time reaches 120. with poisson process"""
    for i in range(n_customer):
        yield env.timeout(random.poisson(1/parameters['lamb']))
        env.process(customer(env, i+1, res_counter, kitchen,parameters, result_fifo))


def startSimulation(n_customer,n_counter,n_kitchen,SIM_TIME,parameters):
    env = simpy.Environment()
    result_fifo = np.zeros((n_customer,len(column_names)))
    counter = Counter(env,n_counter)
    kitchen = Kitchen(env,n_kitchen)
    env.process(customer_arrivals(env,n_customer,counter,kitchen,parameters,result_fifo))
#     env.run(proc)
    env.run(until=SIM_TIME)
    labels = [*range(1,n_customer+1)]
    np_arr = np.array(result_fifo).reshape(n_customer,-1)
    df_fifo=pd.DataFrame(data = np_arr,index=labels,columns=column_names)
    df_fifo=df_fifo.drop(df_fifo[df_fifo.iloc[:,9]==0].index,axis=0) # remove unfinished
    customer
    df_fifo = df_fifo.iloc[:1000]
    total_wait_time = df_fifo.iloc[:,7].sum()
```

```
    total_service_time = df_fifo.iloc[:,8].sum()
    total_time_in_system = df_fifo.iloc[:,9].sum()
    total_counter_idle = df_fifo.iloc[:,6].sum()
    sim_time = df_fifo.iloc[:,5].max()
    return df_fifo,total_wait_time,total_service_time,total_time_in_system,
    counter_total_service_times, total_counter_idle,sim_time
```

Listing 1: fifo.py

```
import simpy
import numpy as np
import pandas as pd
from numpy import random
import math
from helper import round_down
import time


column_names = ["Arrival Time","Current Q Length","Q time","Service Start Time","Food prepare
    start","Exit system time","Food Prepare Duration","Total Wait Time(Queue+Food)","Service
    Time","Total Time in System"]

class Counter(object):
#     Counters to take order
    def __init__(self,env,num_counter):
        self.env = env
        self.counter = simpy.Resource(env,num_counter)
        self.counter_waiting = 0

    def take_order(self,cus,env,service_start,parameters,data):


        print("%s is placing order at counter %.2f" %(cus,service_start))
        time_taken_to_place_order = max(random.exponential(scale = parameters['order_time_mu']),
    parameters['order_time_min'])
        yield self.env.timeout(time_taken_to_place_order)
        service_end = env.now
        print("Order of %s sent to kitchen at %.2f" %(cus, service_end))
        data[cus,8] = round_down(service_end-service_start)
        # if(cus<1000):
        #     counter_total_service_times += service_end-service_start
        # else:
        #     print("stop")
        #     time.sleep(10000)

    def receive_order(self,cus,env,kitchen,parameters,data):

        with kitchen.kitchen.request() as my_turn:
            yield my_turn
            yield env.process(kitchen.prepare_food(cus,env,data,parameters))
            food_end = env.now
            print("%s collected the food at %.2f" %(cus, food_end))
            # Record idle counter and add to total count

class Kitchen(object):
    # Kitchen to prepare food
    def __init__(self,env,num_kitchen):
        self.env = env
        self.kitchen = simpy.Resource(env,num_kitchen)
```

```
48
49      def prepare_food(self,cus,env,data,parameters):
50          print("Kitchen is preparing food for %s at %.2f" %(cus, env.now))
51          food_prepare_time = max(parameters['food_prepare_min'],random.exponential(scale =
        parameters['food_prepare_mu']))
52          data[cus,6] = round_down(food_prepare_time)
53          yield self.env.timeout(food_prepare_time)
54          print("Cooked food for %s at %.2f" %(cus, env.now))
55
56  def customer(env, label, queue, kitchen,parameters, data):
57  #     the customer process arrive at the restaurant and request counter to take order
58      label = label-1
59      arrive_time = env.now
60      print("%s entering the queue at %.2f"%(label,arrive_time))
61  #     data[label,0]=label
62      data[label,0]= round_down(arrive_time)
63      data[label,1] = len(queue.counter.queue)
64      with queue.counter.request() as my_turn:
65          yield my_turn
66          service_start = env.now
67          data[label,3] = round_down(service_start)
68          queue_time = service_start - arrive_time
69          data[label,2]= round_down(queue_time)
70          # placing order at counter
71          yield env.process(queue.take_order(label,env,service_start,parameters,data))
72          # waiting order at counter
73          prepare_food_start = env.now
74          data[label,4] = round_down(prepare_food_start)
75          yield env.timeout(0)
76      yield env.process(queue.receive_order(label,env,kitchen,parameters,data))
77      # prepare_food_end = round_down(env.now)
78      # counter_total_wait_times += round_down(prepare_food_end - prepare_food_start)
79      # receive food from counter
80      exit_time = env.now
81      data[label,5] = round_down(exit_time)
82
83      # total wait time
84      data[label,7] = round_down(data[label,6]+data[label,2])
85      # total time in system
86      data[label,9] = round_down(exit_time-arrive_time)
87
88  # Simlating possion process for customer arrival
89  def customer_arrivals(env,n_customer,res_counter,kitchen,parameters,result_ticket):
90      """Create new *customer* until the sim time reaches 120. with poisson process"""
91      for i in range(n_customer):
92          yield env.timeout(random.poisson(1/parameters['lamb']))
93          env.process(customer(env, i+1, res_counter, kitchen,parameters, result_ticket))
94
95
96  def startSimulation(n_customer,n_counter,n_kitchen,SIM_TIME,parameters):
97      env = simpy.Environment()
98      result_ticket = np.zeros((n_customer,len(column_names)))
99      counter = Counter(env,n_counter)
100     kitchen = Kitchen(env,n_kitchen)
101     env.process(customer_arrivals(env,n_customer,counter,kitchen,parameters,result_ticket))
102     env.run(until=SIM_TIME,)
103 #     env.run(until=proc)
104
105     labels = [*range(1,n_customer+1)]
```

```
106     np_arr = np.array(result_ticket).reshape(n_customer,-1)
107     df_ticket=pd.DataFrame(data = np_arr,index=labels,columns=column_names)
108     df_ticket=df_ticket.drop(df_ticket[df_ticket.iloc[:,-1]==0].index,axis=0) # remove
        unfinished customer
109     df_ticket=df_ticket.iloc[:1000]
110     total_wait_time = df_ticket.iloc[:,7].sum()
111     total_service_time = df_ticket.iloc[:,8].sum()
112     total_time_in_system = df_ticket.iloc[:,9].sum()
113     counter_total_idle_times = (n_counter*SIM_TIME - total_service_time)
114     sim_time = df_ticket.iloc[:,5].max()
115     counter_total_service_times = (df_ticket.iloc[:,4] -  df_ticket.iloc[:,3]).sum()
116     return df_ticket,total_wait_time,total_service_time, total_time_in_system,
        counter_total_service_times, counter_total_idle_times,sim_time
```

Listing 2: ticket.py