CASE STUDY
FOR GRACIOUS UNIVERSITY HOSPITAL

Ibrahim El Shar
Boyuan Lai

University of Pittsburgh
IE Department

Sep 30, 2019

Contents

1 Introduction 3
1.1 Case Overview 3
2 Methodology 3
2.1 Assumptions 3
2.2 Performance Measures 4
3 Results 4
3.1 Experiments 5
4 Critical Assessment 6
5 Recommendation 7
6 Appendex 8

1 Introduction

In this report, we review the case “A Prescription for Budget Woes at Gracious University Hospital”. First, we give an overview of the case and list the questions posed by the case. In an attempt to answer the questions, we develop a simulation model using Python 3.6 and SymPy 1.4 and run several experiments which we base our recommendations on. The Python code used to run all experiments is presented in the appendix.

1.1 Case Overview

The chief financial officer of Gracious University Hospital, Sandra Jones, was notified that the hospital needs to reduce its operational costs by an additional 10% from its last year budget. Jones then has to develop a financial plan to get the hospital back in good standing, overcoming the new financial challenges. Since the issue of pharmacy operations was escalating, Jones has decided to start from pharmacy in her new plan to reduce the operational costs as well as ensuring hospital operations.

Pharmacists have expressed their frustration from spending a significant amount of their time tracking inventory and responding to the floor orders because the automated dispensing cabinets (ADCs) were out of stock. These tasks distracted the pharmacists from their main duties in providing patient care and safety.

To respond to this issue, Jones decided that a physical simulation would be a good initial step. At each floor, the simulation consisted of one physician, two patients, one nurse, one pharmacy technician, one pharmacist, and one timekeeper. The tasks of each of the participants are listed in Table 3 shown in the appendix. For a complete description of the simulation please refer to the case [1].

The main objective of the simulation is to come up with an ADC restocking policy that allows serving as many patients as possible.

2 Methodology

We use an (s, S) policy as the restocking policy for the ADC. In a (s, S), inventory is ordered whenever the inventory level (IL) drop below the reorder point denoted by “s” and the quantity (Q) that will be order is equal to the order-up-to-level denoted by “S” minus the currently inventory level, i.e.,

$$Q = S - IL.$$

Thus at any time the maximum stock level is S.

2.1 Assumptions

Since we are not provided with any knowledge regarding the distribution of the time it takes to do any of the pharmacy activities, we use a uniformal distribution to model the waiting time for each process. This assumption is usually okay in practice since one can usually get the minimum and maximum time estimates information. Our list of assumptions include the following:

1. A 12 hours shift for each replication.
2. The initial inventory level for each shift is equal to the order-up-to-level (S).
3. The time in minutes for the physician to exam a patient follows a uniform distribution $U(5, 15)$.
4. The time in minutes for the nurse to process the prescription, check the ADC for the medications and delivery the medication if available follows a a uniform distribution $U(1, 5)$.
5. In case the medication were not available from the ADC, the pharmacy technician will take a random time, that is uniformly distributed between 40 mins and 1.5 hrs, to restock the ADC from the Central pharmacy. The nurse then takes a random time that is uniformally distributed between 1 and 5 mins to deliver the medication to the patient. That is, we assume the time the nurse needs to administer versus ordering medications has the same distribution.
6. We assume that the nurse will notify the pharmacy technician when any of the two medication inventory level drop below the reorder point. The pharmacy technician will then restock the inventory up to the order-up-to-level taking a random time that is uniformly distributed between 40 mins and 1.5 hrs.
7. The demands for drug A and drug B on each prescription are uniformly distributed between 1 and 6 for drug A and between 2 and 4 for drug B.
8. We assume that a patient leaves a bed only after receiving the medication and that there is always a patient waiting to take an empty bed.

2.2 Performance Measures

The performance measures we consider in this study includes:

1. The number of patients served during each time period/ time shift.
2. The number of times the pharmacy technician needs to restock the ADC (i.e., the number of times he needs to visit the central pharmacy).
3. The ADC space we need for each medication to incorporate the policy.

3 Results

In this section, we conduct several experiments with different (s, S) policies and demand models. We compare the results based on the stated performance measures. We finally conclude with our recommendation of the best strategy based on our findings.

3.1 Experiments

We consider four experiments each of which has a different demand model. The demand models used for each experiment are shown in Table 1. All demand models share the same mean of 3.5 for drug A and 3 for drug B. Table 2 shows the numerical results including the performance measures from running each experiment for 100 replications.

Table 1: Experiments cases with different demand distributions.

Experiment #	Drug A			Drug B		
	Distribution	Min	Max	Distribution	Min	Max
1	Uniform U(1,6)	1	6	Uniform U(2,4)	2	4
2	Beta $\alpha = 3.75,\ \beta = 6.25$	2	6	Beta $\alpha = 6,\ \beta = 2$	0	4
3	Uniform U(1,6)	1	6	Beta $\alpha = 6,\ \beta = 2$	0	4
4	Poisson $\lambda = 3.5$	0	-	Poisson $\lambda = 3$	0	-

Table 2: Simulation results of each experiment for different (s, S) policy.

	s		S		# patients serv.	# visits to C.Ph.	I.L mean (std)		% serv.
	Drug A	Drug B	Drug A	Drug B	mean (std)	mean (std)	Drug A	Drug B	by ADC
Exp.1	55	50	75	70	72.40 (2.58)	15.87 (1.91)	44.12 (2.50)	43.06 (1.90)	100%
	50	45	105	100	72.77 (2.45)	7.51 (0.70)	61.10 (2.90)	53.43 (1.97)	100%
	10	10	35	30	43.77 (3.44)	12.1 (1.14)	10.56 (1.24)	8.63 (0.75)	55%
	55	50	200	180	72.49 (2.52)	1.93 (0.26)	124.88 (4.56)	114.40 (3.78)	100%
Exp.2	55	50	75	70	72.79 (2.72)	15.75 (1.72)	43.97 (3.19)	42.89 (2.50)	100%
	50	45	105	100	73.12 (2.56)	6.87 (0.72)	61.36 (2.83)	60.11 (2.86)	100%
	10	10	35	30	43.19 (3.95)	12.01 (1.21)	10.50 (0.96)	9.18 (0.97)	54%
	55	50	200	180	73.13 (2.29)	1.95 (0.22)	125.94 (4.37)	114.15 (4.32)	100%
Exp.3	55	50	75	70	72.27 (2.61)	15.98 (2.04)	44.48 (2.745)	42.94 (2.00)	100%
	50	45	105	100	72.29 (2.65)	6.91 (0.55)	61.29 (3.00)	59.49 (2.42)	100%
	10	10	35	30	43.08 (3.53)	12.35 (1.15)	10.55 (1.00)	8.74 (0.80)	53%
	55	50	200	180	72.30 (2.34)	1.96 (0.20)	125 (4.60)	114.57 (3.18)	100%
Exp.4	55	50	75	70	72.78 (2.41)	15.94 (2.20)	43.94 (2.20)	42.72 (1.93)	99.89%
	50	45	105	100	73.18 (2.55)	7.14 (0.40)	60.57 (2.14)	59.55 (2.24)	100%
	10	10	35	30	44.16 (3.39)	12.03 (1.21)	10.05 (0.82)	8.56 (0.71)	55%
	55	50	200	180	72.94 (2.51)	1.91 (0.29)	125.30 (4.03)	114.06 (3.27)	100%

We can see from Table 2 that the demand distribution has little to almost no effect on the performance measures as long as all the distributions have the same mean. We discuss this in more details in the next section and for now we focuss on the effects of the policies considered in each experiment.

We investigate four different (s, S) policies. The first policy has a high reorder points $(s_A = 55, s_B = 50)$ relative to the order-up-to-levels $(S_A = 75, S_B = 70)$ for both drugs A and B. This policy performs very well since the mean number of served patients is 72 and demands are served directly from the ADC. Note that if the ADC is always used to serve the patients (i.e., without having the pharmacy technician to visit the central pharmacy due to stockouts) then in this case the physician is the bottleneck of the simulation model since he is the first to see the patients. To see that, note that the average time the physician spend with a patient to write a prescription is 10 mins. So in a period of 12 hrs, the physician can see on average 72 patients which confirms our results. This policy also strikes a good trade-off between the number of patients served, the number of visits to the central pharmacy (15.87 on average) and the mean inventory level which is in this case equal to 44 and 43 for drug A and B respectively.

The second policy we consider has reorder points ($s_A = 50, s_B = 45$) and high order-up-to-levels ($S_A = 105, S_B = 100$). While this policy and the first policy serve the same number of patients on average (also both 100% from ADC) the mean inventory level for both drugs is higher than that of the first policy. Nonetheless, this policy reduces the pharmacy technician visits to the central pharmacy by half compared to the first policy. We do think, however, that it is better to have the technician visit the central pharmacy more often to restock the ADC than having a higher level of inventory. This is due to the fact that a higher inventory level will incur a higher cost on average and will need a larger space requirement to fit more than 100 drugs ($S_A = 105$ and $S_B = 100$).

The third policy has a low reorder points ($s_A = 10, s_B = 10$) and a low order-up-to-levels ($S_A = 35, S_B = 30$). This policy clearly understock inventory ($I.L_A = 10.56$ and $I.L_B = 8.63$) and serves a low number of patients on average (43.77). Nonetheless, only 55% of the patients are served from the ADC.

The last policy has a reorder points ($s_A = 55, s_B = 50$) and a very high order-up-to-levels ($S_A = 200, S_B = 180$). This policy clearly overstock inventory ($I.L_A = 124.88$ and $I.L_B = 114.40$). Moreover, using this policy the technician will have to visit the central pharmacy only about 2 times on average.

Based on the above discussion, we can see that using ($s_A = 55, s_B = 50, S_A = 75, S_B = 70$) we can serve a large number of patients on average mostly from the ADC with an acceptable number of average inventory level and visits to the central pharmacy.

For each of the four experiments, Figures 1 to 4, show a histogram of the number of patients served and the average inventory level of medications A and B during the 12 hrs shift for the first policy. The histograms show the distribution of the number of patients served which peaks at 72 in each experiment. The inventory level plots show the inventory level decrease from the order-up-to-levels of drugs A and B to reach around an $I.L$ of 40 before the next order arrives.

Table 4 shows the 95% confidence intervals for the performance measures of the first policy. The confidence intervals are obtained from the distribution of the simulated data of each experiment.

4 Critical Assessment

In our input parameters we assume uniform distribution for the waiting time for each process. While this assumption keeps our model robust, our simulation outcome and thus our recommendations can be made more specific if more information was provided on the distribution of time of each process.

To study the effect of the input demand distribution on the simulation, the experiments conducted in section 3 has different demand distributions while the mean demand remains the same. From Table 2, we see the performance measures for these four experiments are very close. We can conclude that the mean demand plays an important role while modelling the demand in contrast to the demand distribution which is seen to have little to almost no effect.

It can be noticed that, the percentage of patients served by ADC of Experiment 4 with policy $s = (55, 50)$ and $S = (75, 70)$ is 99.89% while other experiments with this policy are 100%. An explanation of this observation is that, in Experiment 3, the distribution of demand follows a Poisson distribution whose range is unbounded from above while the distribution of demand in other experiments are all bounded.

In the above section, we have discussed the sanity of our simulation model and how an average of 72 patients served seems reasonable considering the fact that in the specific case of all patients being served by ADC, the physician is the bottleneck. This argument serves as an analytical check for our model.

In a real life setting, our simulation model can be further improved by reviewing the model input assumptions with subject metter experts in a model review/white board exercise.

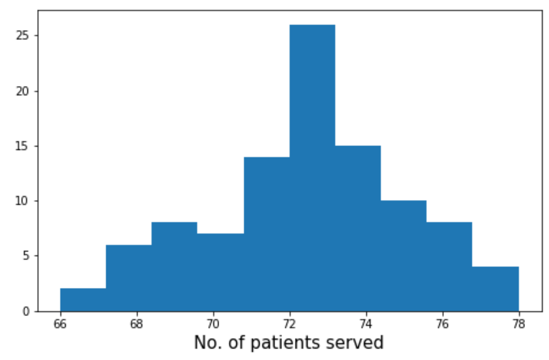
5 Recommendation

During the experiment, we change the (s, S) policy parameters by an increment of 5. Policy 1 is recommended since it strikes a good balance between all performance measures and because all of the patients are served by the ADC in experiments 1-3 while keeping the average number of inventory at an acceptable level. As we increase both the reorder point and the order-up-to-levels from the values of policy 1 the percentage of patients served by the ADC remains 100%.

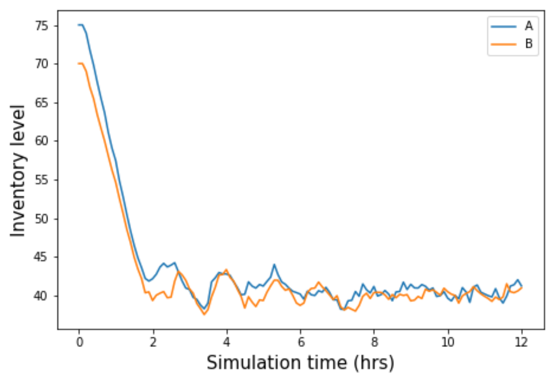
6 Appendex

Table 3: Roles and Responsibilities of Each Participant in the Pharmacy Simulation

Role	Responsibilities
Physician	Listens to patient and writes a prescription corresponding to patients demand
Patient	Visits the physician with his/her demand
	Receives medication from nurse
Nurse	Receives prescription from physician
	Checks automated dispensing cabinet (ADC) to fill prescription
	Delivers medication to patient
Pharmacy technician	Fills requests from the nurse if there is a stockout
	Takes order to Central Pharmacy and waits for it to be filled
	Delivers order to ADC
	Responsible for periodically monitoring the quantity of drug in the ADC and replenishing the ADC according to the restocking policy
Pharmacist	Assembles medication
	Processes refill requests and gives medication to the pharmacy technician
Recorder/ Timekeeper	Keeps track of how long patients wait to receive medication
	Keeps track of nurse and pharmacy technician utilization

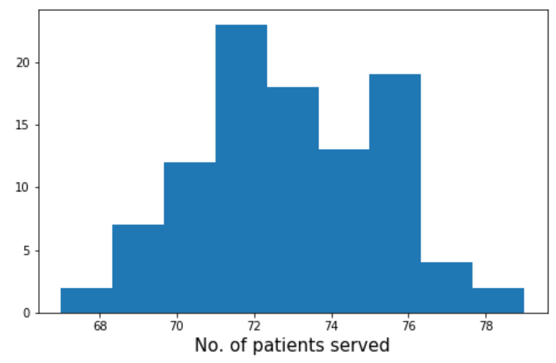


(a) Histogram of No. of patients served

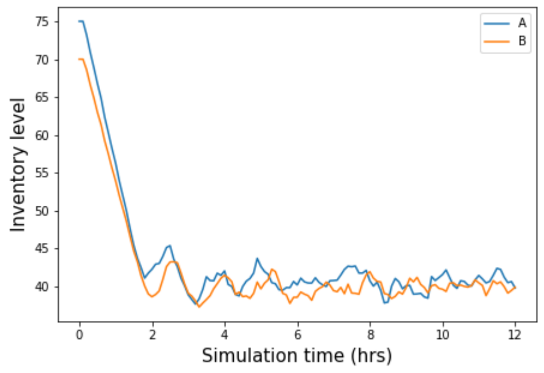


(b) Average inventory level over time

Figure 1: Experiment 1

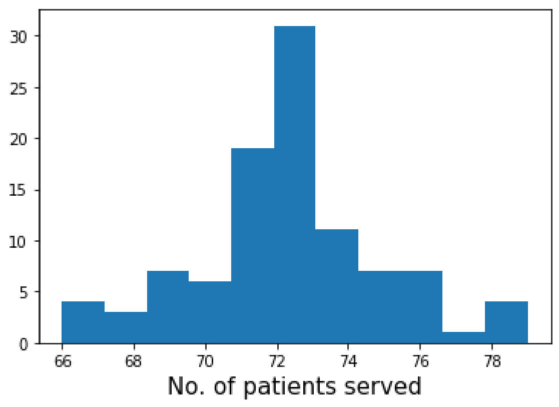


(a) Histogram of No. of patients served

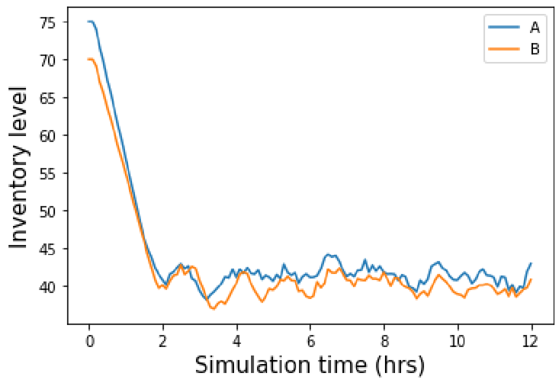


(b) Average inventory level over time

Figure 2: Experiment 2

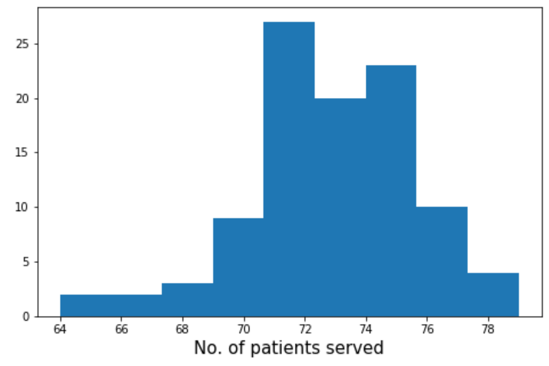


(a) Histogram of No. of patients served

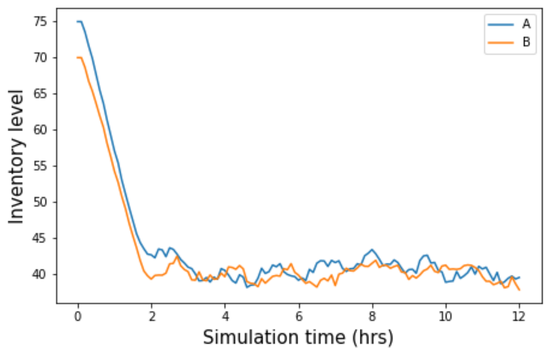


(b) Average inventory level over time

Figure 3: Experiment 3



(a) Histogram of No. of patients served



(b) Average inventory level over time

Figure 4: Experiment 4

Table 4: 95% confidence interval of the performance measures of policy 1

	Experiment 1	Experiment 2	Experiment 3	Experiment 4
No. of patients	(68, 77)	(69, 77)	(67, 78)	(67, 78)
Physician visit C.Ph.	(11.48,18.53)	(11, 19)	(11, 19)	(11.48, 18)
Inventory level of Drug A	(38.10, 48.81)	(39.67, 47.35)	(38.80, 48.96)	(37.41, 49.53)
Inventory level of Drug B	(38.97, 46.39)	(38.67, 45.91)	(39.17, 46.71)	(38.08, 47.13)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import simpy
import numpy as np
import matplotlib.pyplot as plt

class Hospital_simulation(object):
    def __init__(self, env,reorder_pt, order_target):
        self.env = env
        # start initial inventory from the order-upto-levels
        self.inventory = order_target.copy()
        self.num_ordered = np.array([0,0])
        self.patient = 0
        self.reorder_pt = reorder_pt
        self.order_target = order_target
        self.patient_status = [0,0]
        self.demand = np.array([0,0])
        # Start the run process everytime an instance is created.
        self.action = self.env.process(self.serve_patient())
        self.observe_run =self.env.process(self.observe())
        self.flag =0

        self.obs_time = []
        self.inventory_level = []
        self.next_delivery=np.array([float('inf'),float('inf')])
        self.tech_visits_cent = 0
        # track the number of patients not served by ADC
        self.patient_not_serv_by_ADC = 0

    def serve_patient(self,):
        ''' Run simulation'''
        while True:
            nur_flag = 0
            phy_flag = 0
            # Find the patients that are not seen by the physician yet
            k =[i for i,j in enumerate(self.patient_status) if j==0]
            print('Start simulation at %f' % self.env.now)
            print('inventory',self.inventory)
            if len(k)>0:
```

```
            m=k[0]
            phy= self.env.process(self.physician_event(m))
            phy_flag = 1
            print( 'patient_status=', self.patient_status,'at',env.now)
        # Find the patients that are not seen by the nurse yet
        l =[i for i,j in enumerate(self.patient_status) if j==1]
        if len(l)>0:
            m=l[0]
            nur=self.env.process(self.nurse(m))
            nur_flag = 1
            print( 'patient_status=', self.patient_status, 'at', env.now)
        if  phy_flag & nur_flag:
            print('Entered phy_flag & nur_flag')
            yield phy & nur
        elif phy_flag:
            print('Entered phy')
            yield phy
        elif nur_flag:
            print('Entered nur')
            yield nur

def observe(self):
    ''' Keeps track of the I.L in 0.1 time intervals '''
    while True:
       self.obs_time.append(self.env.now)
       self.inventory_level.append(self.inventory.copy())
       yield self.env.timeout(0.1)



def physician(self):
   '''
   Listens to the patient and writes a "prescription" corresponding
   to patient's "demand".
   '''
   self.demand = self.generate_prescription()
   return np.random.uniform(5/60, 15/60)

def physician_event(self,m):
```

```
            self.patient += 1
            duration = self.physician()
            print('phys time',duration, 'happens at', self.env.now + duration)
            yield  self.env.timeout(duration)
            # Changes the status of a patient to seen by the Dr.
            self.patient_status[m]=1
            print( 'patient_status=', self.patient_status,'at', self.env.now)


    def generate_prescription(self):
        ''' Generates the demand for each drug'''
        return np.array([np.random.randint(1,6+1),
                         round(4*np.random.beta(6,2))])

    def nurse(self, k):
        '''
        Receives "prescription" from physician
        Checks automated dispensing cabinet (ADC) to fill prescription
        Delivers "medication" to patient
        '''
        nurse_time = self.nurse_deliver_medication()
        print('nurse time',nurse_time,'happens at', self.env.now + nurse_time)
        print('self.demand:',self.demand)
        yield self.env.timeout(nurse_time)

        flag = 0
        for i in range(2):
            if self.inventory[i] >= self.demand[i]:
                self.inventory[i] -= self.demand[i]
                flag += 1
            else:
                self.inventory[i] = 0

        if flag < 2:
            # if the ADC I.L is not enough to satisfy demand
            self.patient_not_serv_by_ADC += 1
            print("flag < 2 at:", self.env.now)
            if min(self.patient_status) <=1:
                # change the status of the patient to seen by nurse
                # and waiting fot central pharmacy delivery
                self.patient_status[k] = 2
                print('patient_status',self.patient_status,
                      'at:',self.env.now)
                # send technician to the central pharmacy
                self.env.process(self.handle_ADC_restock(k))
                if min(self.patient_status) >1:
                    # if both patient are waiting
                    # the central pharmacy delivery
                    print('pateint_status for both patients is 2')
                    print('env.now:',self.env.now)
                    print('Next delivery time:',min(self.next_delivery))
                    # need to yield the next delivery event
                    # since both patients are waiting fot C.Ph delivery
                    yield self.env.timeout(min(self.next_delivery)-self.env.now)
                    print('After yield (i.e., next delivery event)')
                    self.patient_status = np.array([1,1])
                    print('patient_status',self.patient_status,
                          'at:',self.env.now)
        else:
            # demand can be satisfied from ADC
            # change the patient status to can be seen by the Physician
            # equivalent to receiving a new patient
            self.patient_status[k] = 0
            print('patient_status',self.patient_status, 'at:',self.env.now)
            # check if the ADC I.L is below the reorder pts
            # if yes reorders
            self.env.process(self.handle_ADC_restock(k))
        print('self.patient_status',self.patient_status, 'at:',self.env.now)


    def nurse_deliver_medication(self):
        ''' Generate the time the nure needs to order/give medication'''
        return np.random.uniform(1/60,5/60)


    def handle_ADC_restock(self,kl):
        '''
        Physician technician
        Fills requests from the nurse if there is a “”stockout
        Takes order to Central Pharmacy and waits for it to be filled
        Delivers order to ADC
        Responsible for periodically monitoring the quantity of “”drug
        in the ADC and replenishing the ADC according to the restocking policy
        '''
        print('Physician technician is called to refill ADC')
        idx = [0,0]
        for i in range(2):
            if self.inventory[i] < self.reorder_pt[i] and\
                self.num_ordered[i]== 0:
                idx[i]=1
                self.num_ordered[i] = self.order_target[i] - self.inventory[i]
        # if the IL is below reorder points and there is no ongoing orders
        if max(idx)==1:
            # increment the technician visits to the central pharmacy
            self.tech_visits_cent +=1
            k =[i for i,j in enumerate(idx) if j==1]
            # generate the time for the technician
            var = np.random.uniform(40/60, 90/60)
            # store next delivery arrival time
            self.next_delivery[k]=var + self.env.now
            print('techinician time:',var, 'restock happens at',
                  self.env.now + var,"inventory=",self.inventory)
            yield self.env.timeout(var)
            # add the number ordered to the IL
            self.inventory[k] += self.num_ordered[k]
            # set back the number ordered to zero
            self.num_ordered[k]=0
            self.next_delivery[k]=float('inf')
            print('restock happened at',self.env.now,
                  "inventory=",self.inventory)
            # change patient status back to can be seen by the nurse
            # so it can give the patient the medication
            if self.patient_status[kl]==2:
                self.patient_status[kl]=1

num_rep=100
num_patient_served =[]
```

```
num_tech_cent =[]
avg_il0 = []
avg_il1 = []
ill0 = np.zeros((100,121))
ill1 = np.zeros((100,121))
patient_not_serv_by_ADC =[]
for _ in range(num_rep):
    np.random.seed(_)
    env = simpy.Environment()
    s = Hospital_simulation(env, np.array([55,50]),np.array([75,70]))
    env.run(until=12)
    num_patient_served.append(s.patient)
    num_tech_cent.append(s.tech_visits_cent)
    il0=np.array([i[0] for i in s.inventory_level])
    il1=np.array([i[1] for i in s.inventory_level])
    ill0[_,:]=il0
    ill1[_,:]=il1
    area_il0 = il0*0.1
    area_il1 = il1*0.1
    avg_il0.append(sum(area_il0)/12.0)
    avg_il1.append(sum(area_il1)/12.0)
    patient_not_serv_by_ADC.append(s.patient-s.patient_not_serv_by_ADC)

print('avg num patient served:', np.mean(num_patient_served))
print('std num patient served:', np.std(num_patient_served))
print('avg num tech cent:', np.mean(num_tech_cent))
print('std num tech cent:', np.std(num_tech_cent))
print('avg ilA:', np.mean(avg_il0))
print('std ilA:', np.std(avg_il0))
print('avg ilB:', np.mean(avg_il1))
print('std ilB:', np.std(avg_il1))
print('avg num patient served by ADC:', np.mean(patient_not_serv_by_ADC))
print('std num patient served by ADC:', np.std(patient_not_serv_by_ADC))
plt.hist(num_patient_served, bins='auto')
plt.xlabel('No. of patients served', fontsize=15)
plt.show()
plt.plot(s.obs_time, np.mean(ill0,axis=0),label='A')
plt.plot(s.obs_time, np.mean(ill1,axis=0),label='B')
plt.xlabel('Simulation time (hrs)', fontsize=15)
plt.ylabel('Inventory level', fontsize=15)
plt.legend()
plt.show()
```

References

[1] K. Hicklin, J. S. Ivy, and A. R. Vila-Parrish. Case—a prescription for budget woes at gracious university hospital. INFORMS Transactions on Education, 17(3):110–115, 2017.