دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی صنایع و سیستم‌های مدیریت

عنوان:

تمرین ۱: یک صف و چند خدمت دهنده

نگارندگان:

پدرام پیرو اصفیا-۹۸۲۵۰۰۶

مهدی محمدی-۹۸۲۵۰۴۱

استاد: دکتر عباس احمدی

درس: اصول شبیه سازی

فروردین ۱۴۰۱

فهرست مطالب

صورت مسئله

- در یک سیستم خدماتی چند ایستگاهه موازی مشتریان به طور یکنواخت با زمان های بین دو ورود "۲" الی "۶" دقیقه با احتمالات برابر ۰.۲ وارد می شوند.
- ظرفیت صف، شش نفر است و در صورت تکمیل بودن ظرفیت، مشتری اجازه ورود پیدا نکرده و برگشت می کند.
- مدت زمان خدمتدهی در همه ایستگاه ها به طور یکنواخت بین "۱۲" الی "۲۱" دقیقه با احتمالات برابر ۰.۱ است.
- نمودار کنترلر شبیه سازی و نمودارهای جریان برای پیشامدهای اصلی را ترسیم کنید. شبیه سازی را برای مدت "۷" ساعت انجام دهید.
- تعداد بهینه خدمت دهنده ها را بدست آورید به طوری که «متوسط زمان انتظار» و «تعداد مشتریان بازگشتی» حداقل شده و «درصد اشتغال هر خدمتدهنده» حداکثر شود.

مدل سازی

- متغیر های حالت: *(Q, status(1), status(2), status(3), …, status(I))*

Q: تعداد مشتریان در صف.

status(i): وضعیت خدمت دهنده i ام؛

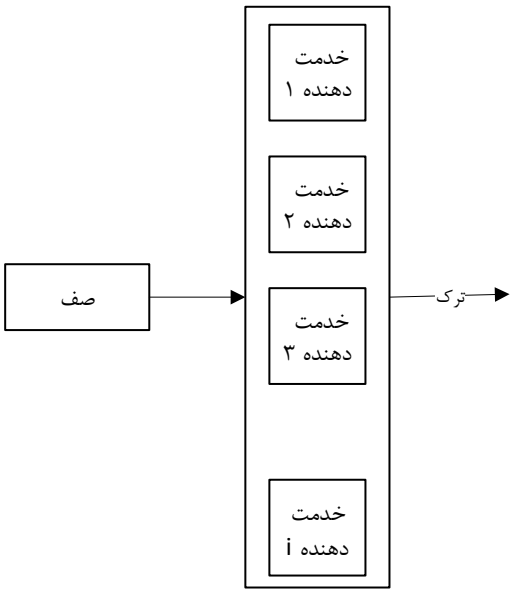
status(i)=0 خدمت دهنده i ام بیکار است.

status(i)=1 خدمت دهنده i ام مشغول است.

- پیشامد های اصلی:

پیشامد ورود : کد ۰

اتمام خدمت‌دهی i: کد i=1,2,…,I

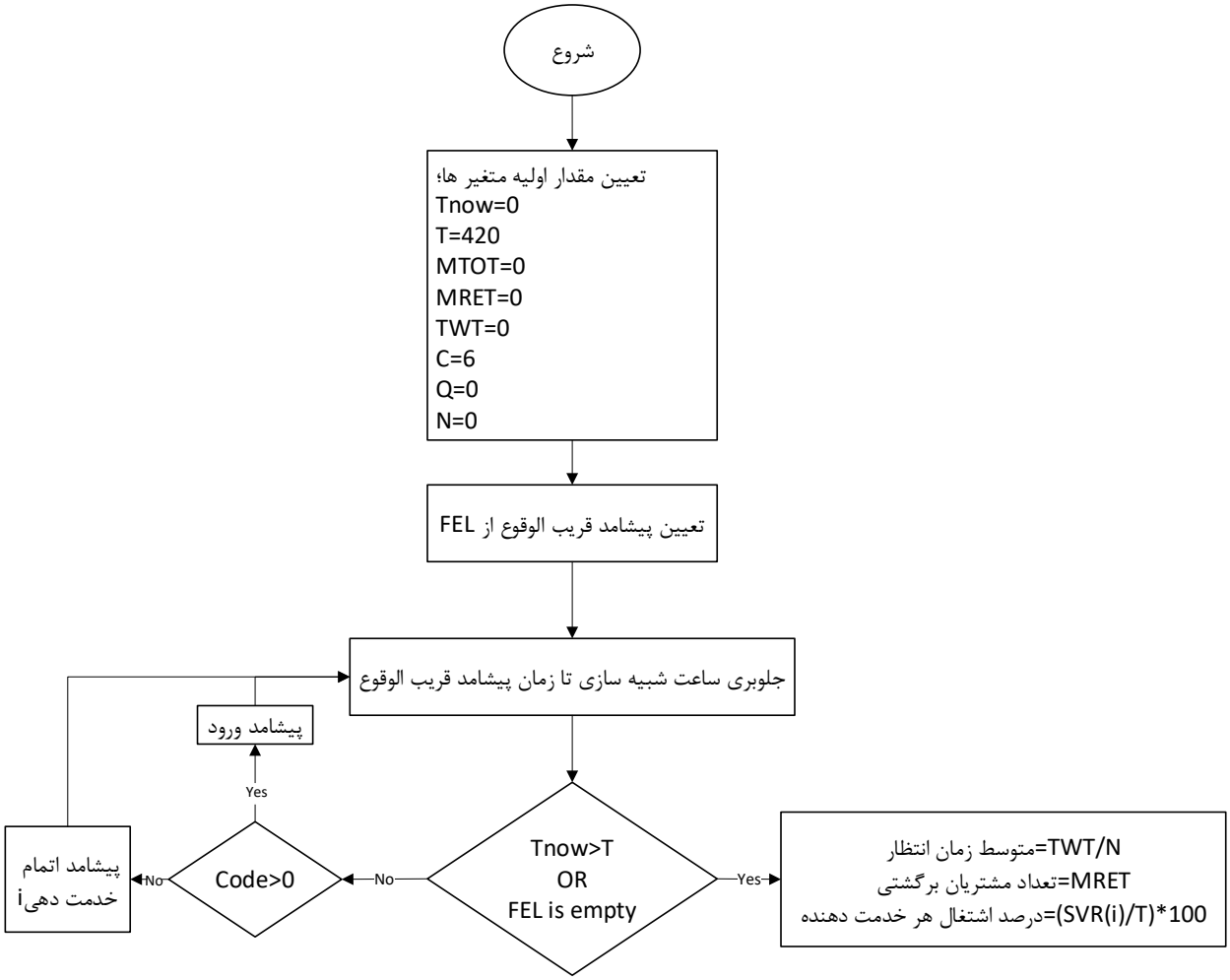


شرح متغیر ها

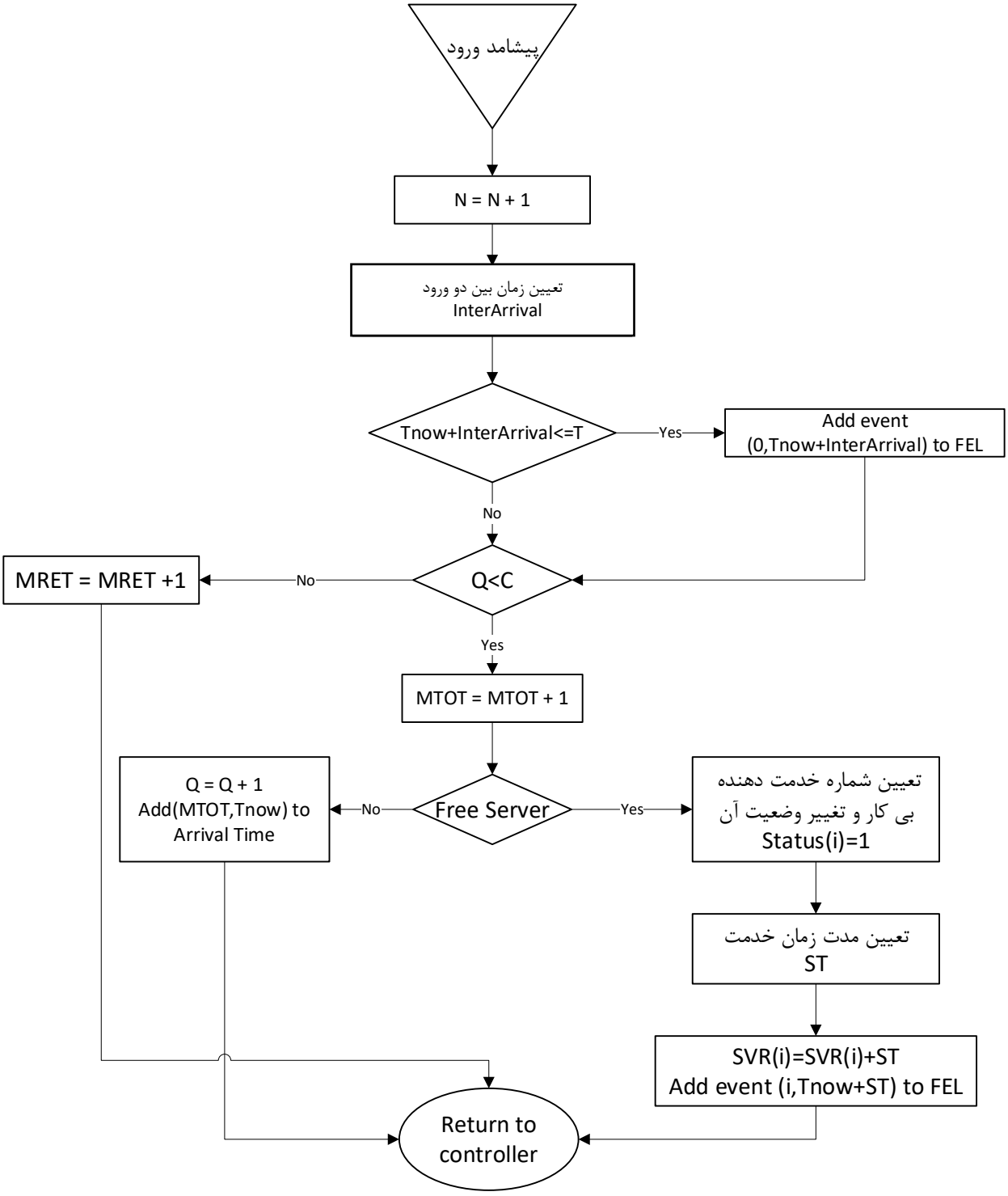
متغیر	شرح
TWT	مجموع زمان انتظار مشتریان
N	تعداد کل مشتریان (شامل این حالت که ممکن است حتی خدمت دریافت نکنند.)
MTOT	تعداد کل مشتریانی که خدمت دریافت می کنند
status(i) = 0 or 1	وضعیت خدمت دهنده **i**
Code=0 or i=1,2,..,I	کد پیشامد
Q	تعداد افراد حاضر در صف
C	حداکثر ظرفیت صف
SVR(i)	زمان اشتغال خدمت دهنده **i**
i	شماره خدمت دهنده
MRET	تعداد مشتریان برگشتی
Tnow	ساعت شبیه سازی
T	مدت شبیه سازی
ArrivalTime(j)	شناسه و زمان وارد شدن **j** امین نفر به صف **(ID , time)**
WT	زمان انتظار مشتری
ST	مدت خدمت دهی
InterArrival	زمان بین دو ورود

lis_MWT	**لیست ذخیره کننده میانگین زمان های ورود به ازای سرور های مختلف**
lis_MRET	**لیست ذخیره کننده تعداد مشتریان بازگشتی به ازای سرور های مختلف**
lis_SVR	**لیست ذخیره کننده درصد زمان اشتغال هر خدمت‌دهنده نسبت به زمان شبیه سازی به ازای سرور های مختلف**
lis_SVR_total	**لیست ذخیره کننده درصد زمان اشتغال هر خدمت‌دهنده نسبت به زمان کل به ازای سرور های مختلف**
FEL	**تمام پیشامد های آینده در این لیست به صورت تاپل های (code,t) ذخیره میشود.**
FEL_backup	**تمامی پیشامد ها از اول تا آخر در این لیست به صورت تاپل های (code,t) ذخیره میشود.**

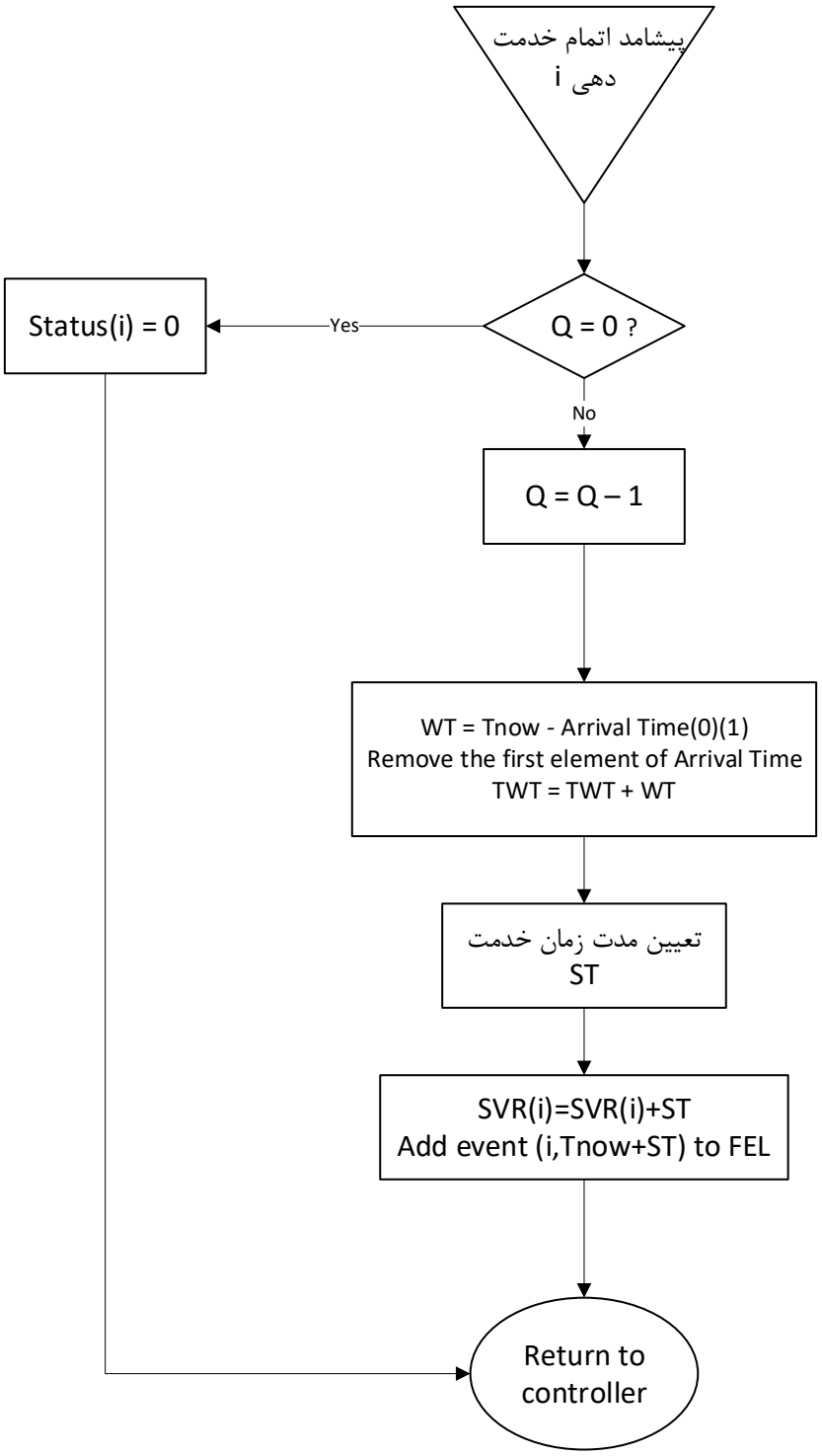
کنترلر شبیه سازی



پیشامد ورود


پیشامد ورود
N = N + 1
تعیین زمان بین دو ورود
InterArrival
Tnow+InterArrival<=T
Yes
Add event
(0,Tnow+InterArrival) to FEL
No
Q<C
No
MRET = MRET +1
Yes
MTOT = MTOT + 1
Free Server
No
Q = Q + 1
Add(MTOT,Tnow) to
Arrival Time
Yes
تعیین شماره خدمت دهنده
بی کار و تغییر وضعیت آن
Status(i)=1
تعیین مدت زمان خدمت
ST
SVR(i)=SVR(i)+ST
Add event (i,Tnow+ST) to FEL
Return to
controller

پیشامد خروج


پیشامد اتمام خدمت
دهی i
Q = 0 ?
Yes
Status(i) = 0
No
Q = Q – 1
WT = Tnow - Arrival Time(0)(1)
Remove the first element of Arrival Time
TWT = TWT + WT
تعیین مدت زمان خدمت
ST
SVR(i)=SVR(i)+ST
Add event (i,Tnow+ST) to FEL
Return to
controller

فرض های ساده ساز

- اولین مشتری در لحظه صفر وارد سیستم می شود.
- به همه ی افراد حاضر در سیستم و صف (حتی اگر زمان شبیه سازی تمام شده باشد) خدمت ارائه می شود.
- در هنگام تخصیص مشتری به خدمت دهنده، از خدمت دهنده اول شروع می کنیم و به اولین خدمت دهنده خالی که برسیم، مشتری را به آن اختصاص می دهیم. (FIFO)

تشریح کد

- ابتدا کتابخانه های مورد نیاز برای دیتافریم،تولید عدد تصادفی و رسم نمودار را وارد می کنیم.

```
import pandas as pd
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
```

- سپس متغیر های مورد نیاز برای ثبت آماره های مسئله را تعریف می کنیم.

```
lis_MWT=[]       #list for mean of waiting times for different number of servers
lis_MRET=[]      #list for total number of returned customers for different number of servers
lis_SVR=[]       #list for total engagement of each server for different number of servers
lis_obj=[]       #list for the value of objective function for different number of servers
lis_SVR_total=[] #list for total engagement of each server for different number of servers
```

- ابتدا یک سید (seed) مشخص می کنیم تا با هربار اجرای کد، برای تحلیل نتایج، نتایج متفاوت نگیریم. باید مسئله را برای تعداد خدمت دهنده های متفاوت شبیه سازی کنیم و سپس آماره ها را با یکدیگر مقایسه کنیم و جواب بهینه را مشخص کنیم. برای این کار از یک حلقه for استفاده میکنیم و برای I های بین ۱ تا ۲۰ مسئله را شبیه سازی می کنیم برای هر I مقدار اولیه متغیر را مشخص می کنیم.

```
np.random.seed(0)
for I in range(1,21):

    '''
       Definition of codes:
       code = 0: Arrival
       code= 1,2,...,I: Departure from ith server
    '''
    TWT = 0                    #total waiting time
    N = 0                      #number of arrival customers (that might have not been given
service)
    MTOT = 0                   #number of customers that have been given service
    Status = np.zeros(I)       #status of i'th server (busy=1 , idle=0)
```

```
    FEL = [(0, 0)]            #(code , time)
    FEL_backup = []           #backup of FEL (saves every event)
    Q = 0                     #number of people waiting in queue
    C = 6                     #capacity of queue
    SVR = np.zeros(I)         #duration of being busy for server i
    MRET = 0                  #number of people returned
    Tnow = 0
    T = 7 * 60                #duration of simulation
    ArrivalTime = []          #arrival time to "Queue" for customers (ID of customer ,
ArrivalTime)
```

- در بدنه قسمت اصلی کد از یک حلقه while استفاده می کنیم و شرط توقف آن به این صورت است که تا زمانی که زمان شبیه سازی کمتر از T است یا FEL خالی نشده تکرار می شود. شرط دوم به این دلیل است که ممکن است بعد از زمان T، افرادی در صف باشند، که آنها نیز خدمت را دریافت کند (جزو مفروضات ساده ساز ذکر شده است). همچنین از یک لیست استفاده می کنیم و تمامی پیشامد ها را در آن ذخیره می کنیم.

```
 while (Tnow <= T) or (len(FEL)!=0):
        FEL_backup.extend(FEL)        #saving every event
```

(تمامی کد های زیر تا جایی که اعلام می شود در ذیل همین حلقه while هستند)

- کد پیشامد قریب الوقوع را بررسی می کنیم اگر صفر بود پیشامد ورود اجرا می شود. در ابتدای پیشامد ورود زمان تا ورود بعدی را مشخص می کنیم و اگر این زمان به علاوه ساعت شبیه سازی بیشتر از T بشود، آن را در FEL وارد نمی کنیم زیرا در غیر این صورت حلقه while بی نهایت می شود؛ سپس بررسی می کنیم که آیا صف به حداکثر ظرفیت خود رسیده است یا خیر، اگر رسیده بود مشتری را بر می گردانیم و در غیر اینصورت به صورت خطی (در مفروضات ذکر شده) سرور ها را بررسی می کنیم و به اولین سرور بیکار که رسیدیم مشتری را تخصیص می دهیم. زمان خدمت را محاسبه می کنیم (برای محاسبات آینده آن را در درایه مربوط به شماره سرور در آرایه SVR ذخیره می کنیم) و پیامد خروج این فرد را وارد FEL می کنیم. در صورتی که سرور بیکار نداشته باشیم، مشتری وارد صف می شود و زمان ورود او به صف به همراه آی دی او (چندمین نفری است که وارد سیستم شده) را ثبت می کنیم.

```
#arrival event
if FEL[0][0]==0:
   N+=1
   InterArrival = np.random.randint(6)+1

   if Tnow+InterArrival <=T :
     FEL.append((0 , Tnow+InterArrival))
```

```
    if Q<C:
       MTOT+=1
       for i in range(I):
           if Status[i]==0:
             Status[i]=1
             ST = np.random.randint(11)+11
             SVR[i]+=ST
             FEL.append((i+1 , Tnow+ST))
             break

          elif i==I-1:
               Q+=1
               ArrivalTime.append((MTOT , Tnow))
               break
       else: MRET+=1
```

- کد پیشامد قریب الوقوع را بررسی می کنیم اگر صفر نبود پیشامد خروج بدین صورت اجرا می شود؛ اگر صف خالی بود سرور i را بیکار می کنیم اگر صف خالی نبود نفر بعدی را از صف وارد می کنیم و با کمک آرایه Arrivaltime زمان انتظار فرد و کل زمان انتظار را محاسبه می کنیم. سپس اولین عنصر آرایه Arrivaltime که مربوط به اولین فرد بود را پاک می کنیم (از لیست خارج می کنیم). زمان خدمت سرور را محاسبه می کنیم (برای محاسبات آینده آن را در درایه مربوط به شماره سرور در آرایه SVR ذخیره می کنیم) و پیشامد خروج فرد را با توجه به زمان محاسبه شده وارد FEL می کنیم.

```
#departure event
else:
     if Q==0:
        Status[FEL[0][0]-1]=0           # server i (or FEL[0][0]) is idle now

     else:     #(Q!=0)
           Q-=1
           WT = Tnow - ArrivalTime[0][1]
           ArrivalTime.pop(0)              #removing the person who is being given service
from ArrivalTime
           TWT+=WT

           ST = np.random.randint(11)+11
           SVR[FEL[0][0]-1]+=ST
           FEL.append((FEL[0][0] , Tnow+ST))
```

- حال اولین عنصر FEL (که الان اجرا شد) را از FEL خارج می کنیم، مجددا FEL را بر اساس زمان پیشامد ها مرتب می کنیم و اگر FEL ما خالی نشده بود ساعت شبیه سازی را به زمان پیشامد قریب الوقوع تغییر می دهیم و مجددا به اول حلقه while می رویم (پایان حلقه while).

```
FEL.pop(0)
FEL.sort(key=lambda x: x[1])
if len(FEL)!=0: Tnow=FEL[0][1]           #updating Tnow
#end of while loop
```

- عناصر تکراری FEL_backup را حذف می کنیم و آن را بر اساس زمان رخ داد ها مرتب می کنیم.

```
FEL_backup = list(set(FEL_backup))
FEL_backup.sort(key=lambda x: x[1]) #sorting all the events chronologically
```

- متغیر های مورد نیاز برای تابع هدف، مقدار تابع هدف و آماره ها را محاسبه می کنیم و در لیست هایی که تعریف کرده بودیم وارد می کنیم. دلیل استفاده از این متغیر ها و همچنین ضرایب تابع هدف در بخش تحلیل نتایج و نتیجه گیری به طور کامل شرح داده شده است.

```
# variables for objective function
MWT = TWT/N*100    #Mean Waiting Time
per_MRET = MRET/N*100    #it is not a statistic (MRET is) but we use this as a value for our
objective function
per_SVR = np.mean(SVR)/Tnow*100  #it is not a statistic (SVR/Tnow ot SVR/T is) but we use this
as a value for our objective function

# Calculating obj function
obj_function = 1*MWT+1.5*per_MRET-1*per_SVR

lis_MWT.append(MWT)
lis_MRET.append(MRET)
lis_SVR.append(SVR/T*100)
lis_SVR_total.append(SVR/Tnow*100)
lis_obj.append(obj_function)
```

- تعداد مشتریان بازگشتی را به ازای هر I (تعداد خدمت دهنده) رسم می کنیم.

```
plt.figure()
plt.style.use('seaborn')
plt.plot(np.arange(1,21) ,lis_MRET , marker='o' )
plt.title('Number of Returned customers')
plt.xlabel('number of servers')
plt.xticks(np.arange(1, 21, 1))
plt.yticks(lis_MRET)
plt.xlim([0,20+0.25])
plt.axhline(0, color='red',linewidth=1)
plt.axvline(0, color='red',linewidth=1)
plt.tight_layout()
plt.show()
```

- میانگین زمان اشتغال سرویس دهنده ها را به ازای هر I (تعداد خدمت دهنده) رسم می کنیم.

```
plt.figure()
plt.style.use('seaborn')
```

```
plt.plot(np.arange(1,21) ,lis_MWT , marker='o' )
plt.title('Mean of waiting time')
plt.xlabel('number of servers')
plt.xticks(np.arange(1, 21, 1))
plt.xlim([0,20+0.25])
plt.axhline(0, color='red',linewidth=1)
plt.axvline(0, color='red',linewidth=1)
plt.tight_layout()
plt.show()
```

- مقدار تابع هدف را به ازای هر I (تعداد خدمت دهنده) رسم می کنیم.مقدار مینیمم این نمودار را مشخص می کنیم.

```
plt.figure()
plt.style.use('seaborn')
plt.plot(np.arange(1,21) ,lis_obj , marker='o' )
plt.plot(np.where(lis_obj==min(lis_obj))[0][0]+1 , min(lis_obj),c='magenta',marker='o')
plt.text(np.where(lis_obj==min(lis_obj))[0][0]+1-1 , min(lis_obj)-80 , s='optimal
point',c='magenta')
plt.title('Least budget objective function')
plt.xlabel('number of servers')
plt.xticks(np.arange(1, 21, 1))
plt.xlim([0,20+0.25])
plt.axhline(0, color='red',linewidth=1)
plt.axvline(0, color='red',linewidth=1)
plt.tight_layout()
plt.show()
```

- I که به ازای آن تابع هدف مینیمم شده را پیدا می کنیم و مقادر آماره های متناظر با آن I را استخراج کرده و در داخل دیتا فریم قرار می دهیم (برای چاپ تمیز تر) و دیتا فریم را نمایش می دهیم.

```
index= np.where(lis_obj==min(lis_obj))[0][0]
df = pd.DataFrame({'Servers':index+1 ,'Capacity of Queue':C ,'Mean of Waiting
Time':lis_MWT[index] , 'Returned Custoemrs':lis_MRET[index],
             'Percentage of Server Engagement (based on Simulation
Time)':np.array_str(lis_SVR[index]),
                  'Percentage of Server Engagement (based on Total
Time)':np.array_str(lis_SVR_total[index])} , index=[0])
df
```

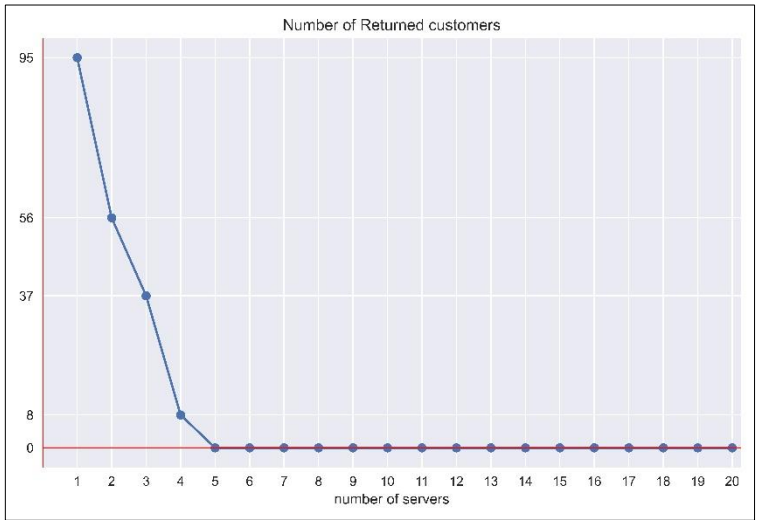
جدول گزارش عملکرد

اسم	فعالیت
پدرام پیرواصفیا	کد پایتون و گزارش (در حد بررسی و اصلاح بعضی جزییات اولیه، تکمیل تحلیل نتایج و نتیجه گیری)
مهدی محمدی	تمامی گزارش‌کار (اعم از صورت مسئله، مدل سازی، شرح متغیرها، کنترل شبیه سازی، پیشامد ورود و خروج،شرح کد، فرض های ساده ساز، نتایج بدست آمده و تحلیل آن)

نتایج بدست آمده

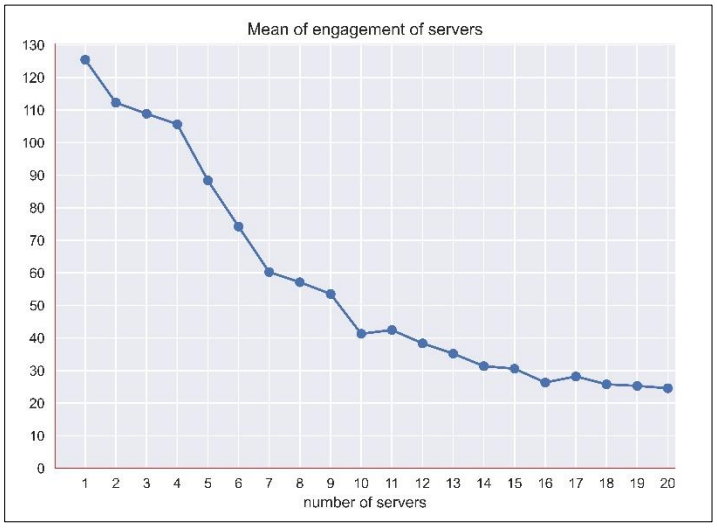
برای اینکه نتایج مدام تغییر نکند و بتوانیم نتایج را صحیح بررسی کنیم، randomseed(0) در نظر میگیریم (توجه داشته باشید در کد خروجی و تحویلی، این بخش از کد، کامنت خواهد شد تا در تصادفی بودن فرآیند شبیه سازی اختلالی پیش نیاید.)

شبیه سازی را به ازای تعداد خدمت دهنده های ۱، ۲، ...، ۲۰ انجام میدهیم و هر بار مقادیر خواسته شده در صورت سوال (تعداد مشتریان بازگشتی، متوسط زمان انتظار، میزان زمان اشتغال هر خدمت دهنده) را جمع آوری میکنیم. نتایج را در نمودار های زیر میتوانید مشاهده کنید:



تصویر ۱: تعداد مشتریان بازگشتی به ازای تعداد خدمت‌دهنده های متفاوت

تصویر ۲: میانگین زمان انتظار به ازای تعداد خدمت‌دهنده های متفاوت



تصویر ۳: میانگین زمان اشتغال به ازای تعداد خدمت‌دهنده متفاوت

تحلیل نتایج و نتیجه گیری

برای تحلیل اطلاعات بدست آمده در بخش قبل و همچنین بدست آوردن تعداد بهینه خدمت‌دهنده، میتوانیم چند حالت در نظر بگیریم:

۱. مشتری مداری و در راس قرار دادن مشتری
۲. حالت میانی (نه هزینه های استخدام سرور زیاد باشد، نه هزینه منتظر ماندن و اذیت شدن مشتریان)
۳. کمبود بودجه برای شروع و استخدام خدمت دهنده کمتر

بررسی تابع هدف برای تعیین و محاسبه تعداد بهینه خدمت دهنده:

برای این منظور، از ۳ آماره مهم استفاده میکنیم:

$$MWT = \frac{TWT}{N} * 100$$

که همان میانگین انتظار هر مشتری است. به دنبال کمینه کردن آن میباشیم.

$$per\_MRET = \frac{MRET}{N} * 100$$

درصد مشتریانی است که بازگشتند و خدمت دریافت نکردند. دلیل اینکه از درصد استفاده میکنیم (مبنای ۰ تا ۱) این است که اگر با "تعداد" کار کنیم باعث میشود تا ضرایب تابع هدف را کوچک تر در نظر بگیریم تا این آماره بیش از حد اهمیت پیدا نکند، برای اینکه دچار این مشکل نشویم و همه چیز در مبنای ۰ تا یک باشد، درصد مشتریان بازگشتی را به عنوان یکی از متغیر های تصمیم انتخاب میکنیم که به دنبال کمینه کردن آن هستیم.

$$per\_SVR = \frac{\overline{SVR}}{Tnow} * 100$$

درصد میانگین اشتغال خدمت دهنده ها. برای محاسبه این آماره، ابتدا از کل زمان اشتغال تمام خدمت دهنده ها (به ازای تعداد سرور مختلف) میانگین میگیریم ($\overline{SVR}$)، سپس بر زمان کل شبیه سازی (نه زمان شبیه سازی اولیه) تقسیم میکنیم (تا حتما بین ۰ تا ۱ شود). توجه داشته باشید، $Tnow$ در آخرین تکرار کد آپدیت میشود و مقدارش برابر با زمان کل شبیه سازی خواهد شد، به همین دلیل از آن میتوان برای محاسبه این متغیر تصمیم استفاده کرد. همچنین به دنبال بیشینه کردن این مقدار هستیم.

پس با این حساب تابع هدف را میتوانیم یک تابع $minimization$ تعریف کنیم:

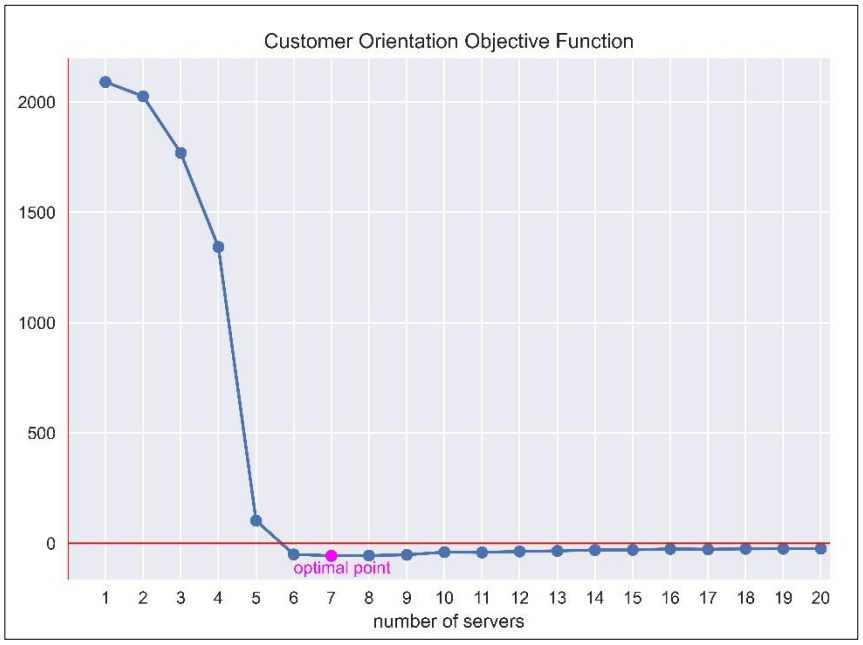
$$Min\ Z = a \times MWT + b \times per\_MRET - c \times per\_SVR$$

بدین قسم که ضرایب $a, b, c \geq 0$ هستند. دلیل اینکه ضریب per_SVR منفی قرار داده شد، این است که باید بیشینه شود (خلاف دو متغیر دیگر).

بررسی حالت اول، مشتری مداری)

در این خصوص، باید مجموع ضرایب مربوط به MWT و per_MRET بیشتر از per_SVR باشد. با توجه به این موضوع که per_MRET مهم تر است (اینکه هیچ مشتری بدون خدمت برنگردد خیلی مهم تر از بقیه

عوامل است.) به همین دلیل $a = 1 , b = 1.5, c = 1$ قرار میدهیم. نتایج تابع هدف را به صورت نمودار میتوان در زیر مشاهده کرد:



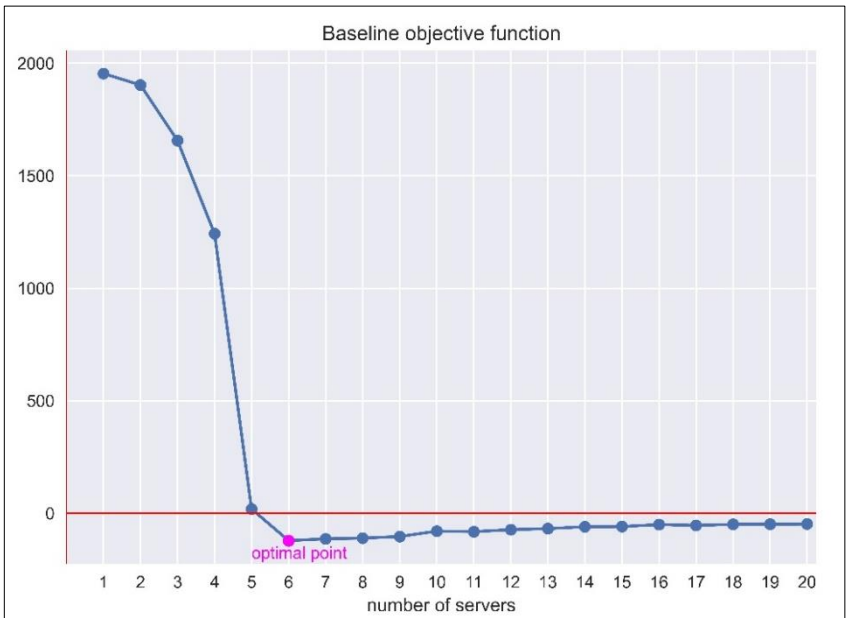
طبق نمودار، تعداد خدمت دهنده های بهینه، ۷ تاست که اطلاعات مربوط به آمارهها را میتوانید در زیر ببینید:

	Servers	Capacity of Queue	Mean of Waiting Time	Returned Custoemrs	Percentage of Server Engagement (based on Simulation Time)	Percentage of Server Engagement (based on Total Time)
0	7	6	1.73913	0	[94.52380952 86.9047619 85.23809524 67.380952...	[90.63926941 83.33333333 81.73515982 64.611872...

متوسط زمان انتظار	1.73
ظرفیت سیستم	6
تعداد خدمت دهنده بهینه	7
تعداد مشتریان برگشتی	0
درصد اشتغال هر خدمت دهنده (بر مبنای زمان شبیه سازی)	[94.52 , 89.90 , 85.23 , 67.38 , 51.66 , 28.80 , 7.14]
درصد اشتغال هر خدمت دهنده (بر مبنای کل زمان شبیه سازی)	[90.64, 83.33, 81.73, 64.61, 49.54 , 27.62, 6.84]

بررسی حالت دوم، حالت میانی)

برای محاسبه این حالت، ضرایب را طوری تعیین میکنیم که مجموع ضرایب اهمیت به مشتری و اهمیت به هزینه سازمان یکسان شود: $a = b = \frac{c}{2} = 1$.



تصویر ۵: تابع هدف حالت بینابین

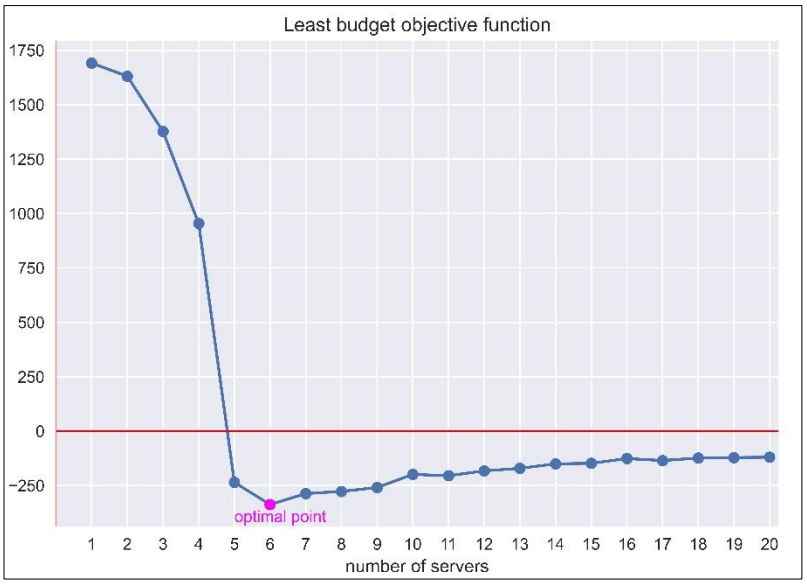
طبق نمودار، تعداد خدمت دهنده های بهینه، ۶ تاست که اطلاعات مربوط به آماره‌ها را میتوانید در زیر ببینید:

	Servers	Capacity of Queue	Mean of Waiting Time	Returned Custoemrs	Percentage of Server Engagement (based on Simulation Time)	Percentage of Server Engagement (based on Total Time)
0	6	6	21.848739	0	[92.38095238 90.47619048 80.95238095 81.428571...	[89.40092166 87.55760369 78.34101382 78.801843...

متوسط زمان انتظار	21.84
ظرفیت سیستم	6
تعداد خدمت دهنده بهینه	6
تعداد مشتریان برگشتی	0
درصد اشتغال هر خدمت دهنده (بر مبنای زمان شبیه سازی)	[92.38, 90.47, 80.95, 81.42, 64.52, 35.71]
درصد اشتغال هر خدمت دهنده (بر مبنای کل زمان شبیه سازی)	[89.40 , 87.55 , 78.34 , 78.80 , 62.44 , 34.56]

بررسی حالت سوم، حالت کمترین بودجه)

برای محاسبه این حالت، ضرایب را طوری تعیین میکنیم که ضرایب اهمیت به اشتغال سرور ها بیشتر از دو متغیر دیگر باشد. به این دلیل که ما میخواهیم حتی الامکان سرور های موجود خیلی کار کنند تا از استخدام بی منظور خدمت دهنده خود داری کنیم. بدین منظور $a = 1, b = 1, c = -5$



تصویر ۶: تابع هدف برای کمترین بودجه

طبق نمودار، تعداد خدمت دهنده های بهینه، ۶ تاست که اطلاعات مربوط به آماره‌ها را میتوانید در زیر ببینید:

	Servers	Capacity of Queue	Mean of Waiting Time	Returned Custoemrs	Percentage of Server Engagement (based on Simulation Time)	Percentage of Server Engagement (based on Total Time)
0	6	6	21.848739	0	[92.38095238 90.47619048 80.95238095 81.428571...	[89.40092166 87.55760369 78.34101382 78.801843...

متوسط زمان انتظار	21.84
ظرفیت سیستم	6
تعداد خدمت دهنده بهینه	6
تعداد مشتریان برگشتی	0
درصد اشتغال هر خدمت دهنده (بر مبنای زمان شبیه سازی)	[92.38, 90.47, 80.95, 81.42, 64.52, 35.71]
درصد اشتغال هر خدمت دهنده (بر مبنای کل زمان شبیه سازی)	[89.40 , 87.55 , 78.34 , 78.80 , 62.44 , 34.56]

نکته عجیب و قابل تامل این است که با اینکه ضریب تابع هدف برای per_SVR بیشتر از دو متغیر دیگر است، جواب بدست آمده با جواب حالت میانی یکسان است. هرچند این سیستم با اینکه مشتری برگشتی ندارد، ولی ایراد اساسی که دارد این است که زمان منتظر ماندن در صف خیلی بالاست و ممکن است باعث نارضایتی مشتری شود و به مرور زمان مشتریان را از دست بدهد. ولی از آنجایی که هم حالت میانی و هم حالت کمترین بودجه به یک نتیجه ختم شده اند، شاید منطقی ترین کار این باشد که تعداد سرور را ۶ تا در نظر بگیریم تا هم کمترین بودجه را خرج کنیم و هم رضایت مشتری تا حدودی ارضا شود.

نکته مهم دیگری که باید به آن اشاره کرد و در هر ۳ نوع حالت وجود داشت، یکسان نبودن میزان اشتغال خدمت‌دهندگان است. این موضوع به این دلیل است که در طی حل سوال و الگوریتم حل، ما به دنبال اولین خدمت دهنده ای میگشتیم که بیکار است (طبق شماره خدمت دهنده) و این باعث میشد که خدمت دهنده های با شماره کمتر، خود به خود بیشتر کار کنند. برای رفع این مشکل میتوانیم هر دفعه لیستی از خدمت دهنده های بیکار پیدا کنیم و به صورت رندوم یکی از آن ها را انتخاب کرده و کار را به وی واگذار کنیم. در این صورت میزان اشتغال همه یکسان حدودا یکسان خواهد بود و انصاف بهتر رعایت خواهد شد.

و مورد آخر اینکه اختلاف مشتری مداری و کمترین بودجه تنها در استخدام یک خدمت‌دهنده است. یعنی با استخدام فقط یک خدمت دهنده بیشتر میتوانیم رضایت مشتری را چند برابر کنیم، چرا که میانگین زمان انتظارشان در این حالت برابر با ۱.۷۳ دقیقه خواهد بود که به مراتب از ۲۱.۸۴ دقیقه بهتر است. این رویکرد باعث میشود به مرور زمان، مشتریان، ما را به آشنایان و اطرافیان خود معرفی کنند و تعداد مشتریان افزایش یابد که باعث افزایش نرخ بازگشت سرمایه میشود و اینجاست که استخدام یک خدمت دهنده بیشتر، خیلی بیشتر به سودمان تمام می شود تا اینکه در هزینه استخدام وی صرفه جویی کنیم.

نتیجه گیری نهایی

پس تعداد خدمت دهنده پیشنهادی ما برای این مسئله و با در نظر گرفتن این متغیر های تصمیم، ۷ تا میباشد.

متوسط زمان انتظار	1.73
ظرفیت سیستم	6
تعداد خدمت دهنده بهینه	7
تعداد مشتریان برگشتی	0
درصد اشتغال هر خدمت دهنده (بر مبنای زمان شبیه سازی)	[94.52 , 89.90 , 85.23 , 67.38 , 51.66 , 28.80 , 7.14]
درصد اشتغال هر خدمت دهنده (بر مبنای کل زمان شبیه سازی)	[90.64, 83.33, 81.73, 64.61, 49.54 , 27.62, 6.84]