ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC KỸ THUẬT MÁY TÍNH

ĐÁNH GIÁ HIỆU NĂNG HỆ THỐNG

BÀI TẬP LỚN

Mô phỏng hệ thống hàng dùng Simpy

Chủ đề 8

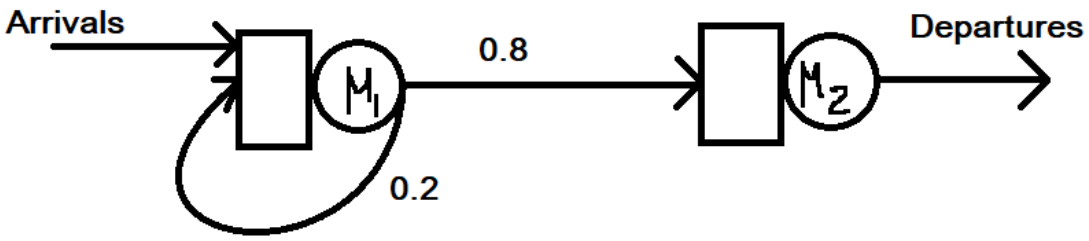
GVHD: **Trần Văn Hoài**
Thành viên : Đặng Hữu Nam - 1720034
Võ Nguyễn Phi Long - 1712033
Từ Nguyên Gia Khiêm - 1711754

TP. HỒ CHÍ MINH, THÁNG 12/2019

Mục lục

1 Giới thiệu

Chủ đề 8:
Mạng 2 hàng $Q_1 = M/M(\mu_1)/1$, $Q_2 = M/M(\mu_2)/1$, trong đó $Q_1 \to Q_2$ với $p_{12} = 0.8$; $Q_1 \to Q_1$ với $p_{11} = 0.2$; quá trình đến Q_1 với λ, công việc sau khi qua Q_2, sẽ rời khỏi hệ thống.



2 Quá trình đánh giá hiệu năng hệ thống

2.1 Xác định các mục tiêu và Định nghĩa Hệ thống

- **Mục tiêu:** Mô phỏng hoạt động của Hệ thống hàng bằng Simpy trong một khoảng thời gian nhất định và Đánh giá độ chính xác của quá trình mô phỏng so với Lý thuyết.
- **System:** Một hệ thống hàng gồm 2 hàng đơn M/M/1 Q_1 và Q_2; mỗi hàng đợi đều có riêng service rate (μ_1) và (μ_2); Job sau khi rời khỏi Q_1 hoặc quay trở lại Q_1 với $p_{11} = 0.2$ hoặc tới Q_2 với $p_2 = 0.2$ sau đó rời khỏi hệ thống.

2.2 Các Chức năng và Kết quả có thể thu được của Hệ thống

- **Chức năng:**
  - Đưa một Job vào trong System
  - Thực hiện Job
  - Kết thúc và Đưa Job rời khỏi hệ thống
- **Các kết quả có thể xảy ra::**
  - Job được thực hiện thành công và rời khỏi hệ thống
  - Job đang chờ trong Q_1
  - Job đang chờ trong Q_2
  - Job vẫn đang được thực hiện trong Q_1
  - Job vẫn đang được thực hiện trong Q_2

2.3 Lựa chọn Metrics:

- Hiệu suất sử dụng hệ thống (Utilization)
- Thời gian phục vụ trung bình của hệ thống (Mean service time)
- Số lượng Job trung bình nằm trong hệ thống (Mean Job in System)
- Thời gian đợi trung bình của các Job được đã được thực hiện (Mean Waiting time of Jobs)
- Phương sai của số lượng Job trung bình nằm trong hệ thống (Variance of Job in System)

2.4 Danh sách các Parameters

2.4.1 Parameter của Hệ thống

- **Service rate của Server 1** : μ_1
- **Service rate của Server 2** : μ_2

2.4.2 Parameter của Tải công việc

- **Arrival rate của Server 1** : λ_1
- **Tổng đơn vị thời gian mô phỏng** : MAXTIME
- **Số lần lấy mẫu cho Transient Remove và Terminating Simulation** : REPS
- **Số lượng Job N để tính xác suất có nhiều hơn N Job trong hệ thống** : ProN
- **Knee được xác định bởi Tan= delta(lenght of queue)/delta(time)** : kneePos
- **Độ tin cậy 1-alpha** : alpha
- **Nguyên tắc quản lý các Job trong queue** : SERVICEDISCIPLINE
- **Số Job tối đa có thể phục vụ** : POPULATION

2.5 Các nhân tố cần khảo sát

- **Service rate của Server 1** : μ_1
- **Service rate của Server 2** : μ_2
- **Arrival rate của Server 1** : λ_1
- **Tổng đơn vị thời gian mô phỏng** : MAXTIME
- **Số lần lấy mẫu cho Transient Remove và Terminating Simulation** : M
- **Số lượng Job N để tính xác suất có nhiều hơn N Job trong hệ thống** : N
- **Knee được xác định bởi Tan= delta(lenght of queue)/delta(time)** : kneePos
- **Độ tin cậy 1-alpha** : alpha

2.6 Lựa chọn kỹ thuật đánh giá

- Kĩ thuật đánh giá :

- Mô phỏng bằng ngôn ngữ lập trình Python thông qua Simpy
- Validation
- **Transient Remove:** Initial Data Deletion
- **Terminating Simulation:** Independent Replications

2.7 Lựa chọn Workload

- Arrival rate λ
- JobGenerator với arrival time là phân phối mu
- Số lần lấy mẫu M
- Tổng đơn vị thời gian lấy mẫu : MAXTIME

2.8 Design Experiments

- Mô phỏng được thiết kế bằng ngôn ngữ python:

2.8.1 Danh sách các package được sử dụng

```
import simpy
#import random
import numpy.random as random
import scipy.stats as ss
import math
import matplotlib.pyplot as plt
```

2.8.2 List parameter

```
''' ------------------------ '''
''' Parameters             '''
''' ------------------------ '''
'Max Simulation Time'
MAXSIMTIME = 5000
' "True" Print to terminal'
VERBOSE = False
'Queue Changing Rate'
P11 = 0.2
P12 = 1-P11
'Original Arrival & Service Rate'
LAMBDA = 9.0
MU1 = 12.0
MU2 = 12.0
'Fixed Arrival Rate'
'Lambda1=10'
'Lambda2=8'
LAMBDA1 = LAMBDA/P12
LAMBDA2 = LAMBDA
'------------'
POPULATION = 50000000
SERVICE_DISCIPLINE = 'FIFO'
'Open file to Write or Not'
LOGGED = True
'Probability Job Queue has more than ProN job(s)'
ProN = 4.0
'Number of Replications'
REPS = 10
'Identify the knee'
kneeDeltaRate = 0.00005
INTER_SERVICE_TIME_1 = 1/MU1
INTER_SERVICE_TIME_2 = 1/MU2
'Convenience Level'
Confidence = 0.95
alpha = 1 - Confidence
```

2.8.3 Định nghĩa Job: Job

```
class Job:
def __init__(self, name, arrtime, duration):
    self.name = name
    self.arrtime = arrtime
    self.duration = duration

def __str__(self):
    return '%s at %d, length %d' %(self.name, self.arrtime, self.duration)

def SJF( job ):
    return job.duration
```

2.8.4 Định nghĩa Server1:

```
def MeanJobInSystemComputing(self, k):
    i = k
    JobxTime = 0
    while i < len(self.queuetime) - 1:
           JobxTime += self.systemJob[i] * (self.queuetime[i+1] - self.queuetime[i])
           i += 1
    self.MeanJobInSystem = JobxTime/(MAXSIMTIME - self.queuetime[k])
def VarOfJobInSystemComputing(self, k):
    i = k
    SquaredDeltaJobxTime = 0
    while i < len(self.queuetime) - 1:
           SquaredDeltaJobxTime += ((self.systemJob[i] - self.MeanJobInSystem)**2) *
               (self.queuetime[i+1] - self.queuetime[i-1])
           i += 1
    self.VarJobInSystem = SquaredDeltaJobxTime/(MAXSIMTIME - self.queuetime[k])
def ProbMoreThanNjobsComputing(self, N, k):
    i = k
    totalTime = 0
    while i < len(self.queuetime) - 1:
        if self.systemJob[i] > N:
            totalTime += self.queuetime[i+1] - self.queuetime[i-1]
        i += 1
    self.ProbMoreThanNjobs = totalTime/(MAXSIMTIME - self.queuetime[k])
```

2.9 Analyze and Interpret Data

2.9.1 Start simulation

```
''' start SimPy environment '''
''' kneePos is the Position of knee defined by transient remove'''
''' we first simulate and once again after transient remove to find difference '''
env = simpy.Environment()
MyServer2 = Server2( env, SERVICE_DISCIPLINE )
```

```
MyServer = Server( env,MyServer2, SERVICE_DISCIPLINE,MU1 ,MU2 )
MyJobGenerator = JobGenerator( env, MyServer, POPULATION, LAMBDA, MU1 )
''' start simulation '''
env.run(MAXSIMTIME)
MyServer.MeanJobInSystemComputing( kneePos)
MyServer.VarOfJobInSystemComputing( kneePos)
MyServer.ProbMoreThanNjobsComputing( ProN, kneePos)
MeanJob = MyServer.MeanJobInSystem
VarJob = MyServer.VarJobInSystem
ProbJob = MyServer.ProbMoreThanNjobs
```

2.9.2 Transient removal

```
'''Transient Remove'''
EnvReps = list(())
MyServer2 = list(())
MyServer = list(())
MyJobGeneration = list(())
MeanJList = list(())
MeanLList = list(())
RateMeanList    = list(())
MeanTimeLine   = list(())
TerminateList = list(())
ServerTimeLen = list(())
MeanOverallRun = 0.0
MeanOverall = 0.0
MeanLdel = 0.0
i = 0
while(i < REPS):
    EnvReps.append(simpy.Environment())
    MyServer2.append(Server2( EnvReps[i], SERVICE_DISCIPLINE ))
    MyServer.append(Server( EnvReps[i],MyServer2[i], SERVICE_DISCIPLINE,MU1 ,MU2 ))
    MyJobGeneration.append(JobGenerator( EnvReps[i], MyServer[i], POPULATION, LAMBDA, MU1 ))
    env = EnvReps[i]
    env.run(until = MAXSIMTIME)
    ServerTimeLen.append(len(MyServer[i].queuetime))
    '''print('%d' %thisRepsQueueLen)'''
    j = 0
    timeRun = 0
    timesRun = 1
    lengthSumRun = 0.0
    TerminateSumRun = 0.0
    while(j < ServerTimeLen[i]):
        if(MyServer[i].queuetime[j] >= timeRun + 1) :
            if(i > 0):
                MeanJList[timeRun] += lengthSumRun / (timesRun*REPS)
            else:
                MeanJList.append(lengthSumRun / (timesRun*REPS))
            lengthSumRun = MyServer[i].queuelength[j]
            timeRun += 1
            timesRun = 1
        else:
```

```
            lengthSumRun += MyServer[i].queuelength[j]
            timesRun += 1
        TerminateSumRun += MyServer[i].queuelength[j]
        j += 1
    TerminateList.append(TerminateSumRun)
    if(i > 0):
        MeanJList[MAXSIMTIME - 1] += lengthSumRun / (timesRun*REPS)
    else:
        MeanJList.append(lengthSumRun / (timesRun*REPS))
    i += 1

'''-----------OPEN FILE TO PRINT------------'''
if LOGGED:
    qlog = open( 'mm1-l%d-m%d.csv' % (LAMBDA,MU1), 'w' )
    qlog.write( '0\t0\t0\n' )
'''-----------MEAN OVERALL------------'''
i = 0
while i < MAXSIMTIME :
    MeanOverallRun += MeanJList[i]
    i += 1
'----------------------------------------'
MeanOverall = MeanOverallRun*1.0/MAXSIMTIME
qlog.write('%f\n' %(MeanOverall))
'---MEAN DELETE FIRST L OBSERVATION(S)---'
i = 0
setKnee = True
while i < MAXSIMTIME - 1 :
    MeanOverallRun -= MeanJList[i]
    MeanLList.append(MeanOverallRun/(MAXSIMTIME - i))
    RateMeanList.append((MeanLList[i] - MeanOverall)/MeanOverall)
    MeanTimeLine.append(i)
    qlog.write('%f\t' %(MeanJList[i]))
    qlog.write('%f\t' %(MeanLList[i]))
    qlog.write('%f\n' %(RateMeanList[i]))
    if(setKnee):
        if((i > 0) & (abs(RateMeanList[i] - RateMeanList[i - 1]) < kneeDeltaRate)) :
            qlog.write('%f\n' %(abs(RateMeanList[i] - RateMeanList[i - 1])))
            kneePos = i
            setKnee = False
    i += 1
print('Knee: %d\n\n' %kneePos)
plt.plot(MeanTimeLine, RateMeanList)
plt.xlabel('Time')
plt.ylabel('RateMean')
plt.show
```

2.9.3 Show outcome

```
print( 'Queue 1')
print( 'Mean Jobs in System                : %f' % (MeanJob) )
print( 'Variance of Mean Jobs in System    : %f' % (VarJob) )
print( 'Probability System has More than4 Jobs : %f' % (ProbJob) )
```

```
''' print statistics '''
RHO1 = LAMBDA1/MU1
RHO2 = LAMBDA2/MU2
print( 'Arrivals            : %d' % (MyServer.jobsDone) )
print( 'Utilization         : %.2f/%.2f'
    % (1.0-MyServer.idleTime/MAXSIMTIME, RHO1) )
print( 'Mean waiting time   : %.2f/%.2f'
    % (MyServer.waitingTime/MyServer.jobsDone, RHO1**2/((1-RHO1)*LAMBDA1) ) )
print( 'Queue 2')
print( 'Arrivals            : %d' % (MyServer2.jobsDone) )
print( 'Utilization         : %.2f/%.2f'
    % (1.0-MyServer2.idleTime/MAXSIMTIME, RHO2) )
print( 'Mean waiting time   : %.2f/%.2f\n\n'
    % (MyServer2.waitingTime/MyServer2.jobsDone, RHO2**2/((1-RHO2)*LAMBDA2) ) )
```

2.10 Present Results

System Workload Parameter.

0.2		p12:	0.8
Lamda 1	**Lamda 2**	**Muy 1**	**Muy 2**
2.5	2	12	12
3.75	3	12	12
5	4	12	12
6.25	5	12	12
7.5	6	12	12
8.75	7	12	12
10	8	12	12
11.25	9	12	12
12.5	10	12	12

On Computer

```
Mean Jobs in System                     : 3.523665
Variance of Mean Jobs in System         : 20.453968
Probability System has More than4 Jobs  : 0.525767
Arrivals                : 41562
Utilization             : 0.76/0.68
Mean waiting time       : 0.30/0.19
Arrivals                : 30234
Utilization             : 0.43/0.43
Mean waiting time       : 0.05/0.05


Knee: 1


Confidence Interval: [ 2.780772 - 0.234668 : 2.780772 + 0.234668 ]
Mean Jobs in System                     : 3.116847
Variance of Mean Jobs in System         : 17.009794
Probability System has More than4 Jobs  : 0.462549
Arrivals                : 41750
Utilization             : 0.76/0.68
Mean waiting time       : 0.25/0.19
Arrivals                : 30324
Utilization             : 0.43/0.43
Mean waiting time       : 0.05/0.05
```

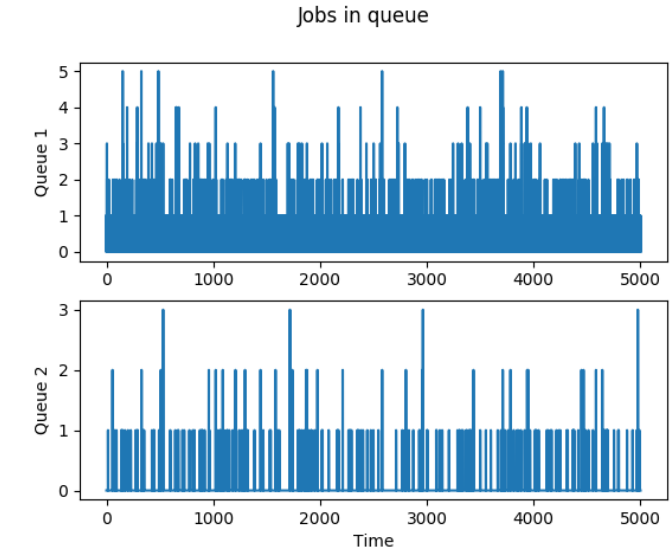
Queue 1

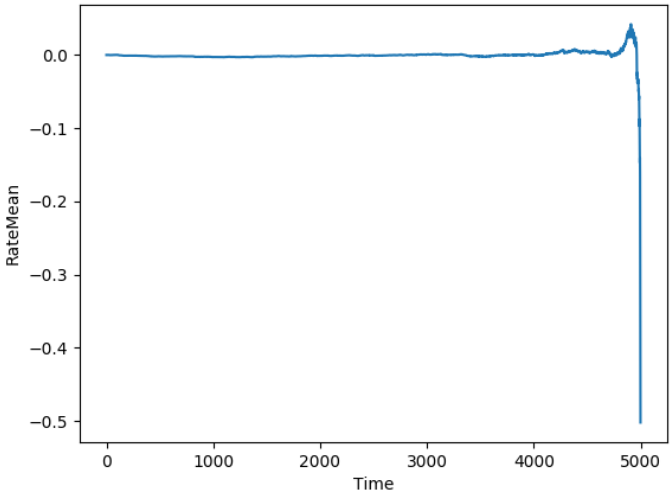
Queue 1							
MeanJob	Var meanjob	P > 4job	Arival	Utilization	Utilization (ly thuyet)	Mean waiting time	Mean waiting time ly thuyet
1.070446	0.221137	0.001562	13832	0.23	0.21	0.03	0.02
1.191006	0.798326	0.012578	20604	0.35	0.31	0.05	0.04
1.357878	1.565541	0.033865	27246	0.45	0.42	0.07	0.06
1.792094	4.774158	0.131386	34518	0.58	0.52	0.11	0.09
2.545629	11.826647	0.321023	41259	0.69	0.62	0.19	0.14
4.060817	39.263547	0.616626	47522	0.79	0.73	0.32	0.22
11.6654263	271.66106	1	54773	0.92	0.83	0.96	0.42
1042.33025	538063	1	59610	1	0.94	81.15	1.25
3041.89893	5612818	1	60107	1	1.04	219.81	-2.08

Queue 2

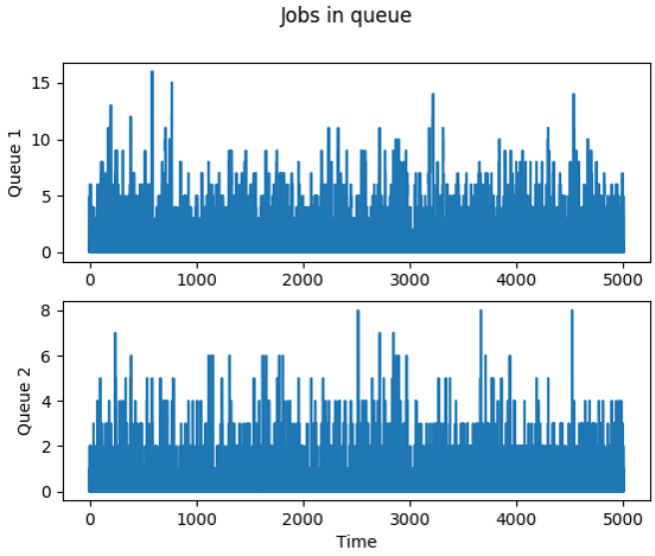
Queue 2				
Arival	Utilization	Utilization (ly thuyet)	Mean waiting time	Mean waiting time ly thuyet
	0.17	0.17	0.02	0.02
149720.25	0.25	0.03	0.03	
19998	0.33	0.33	0.04	0.04
25129	0.42	0.42	0.06	0.06
29965	0.5	0.5	0.8	0.8
34827	0.58	0.58	0.12	0.12
39821	0.66	0.67	0.17	0.17
43212	0.72	0.75	0.22	0.25
43737	0.73	0.83	0.22	0.42

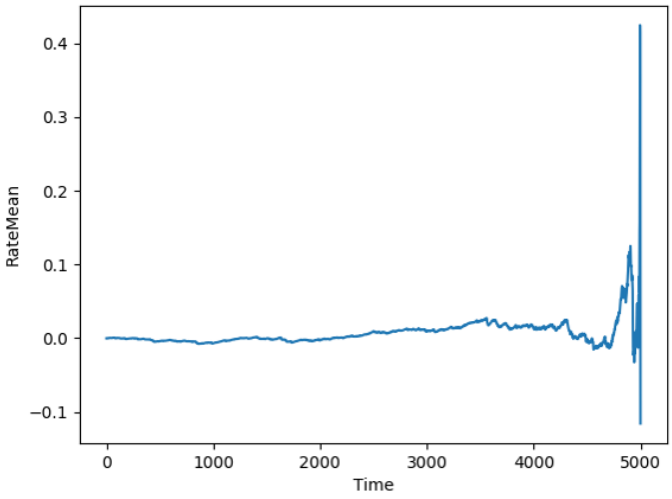
Lambda = 2



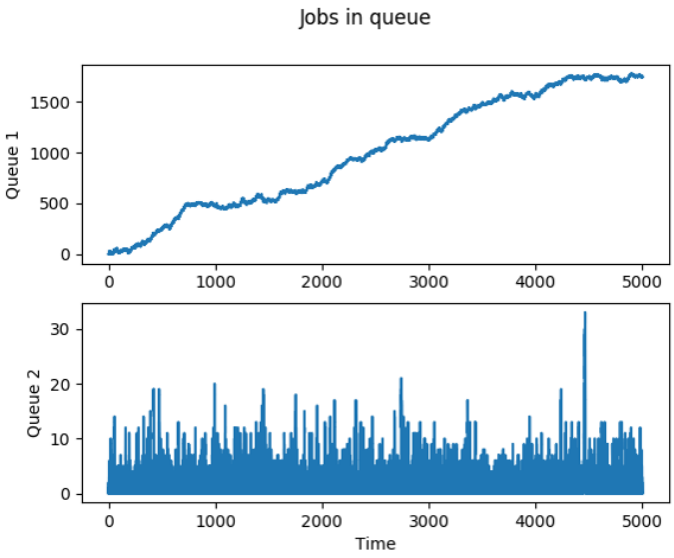


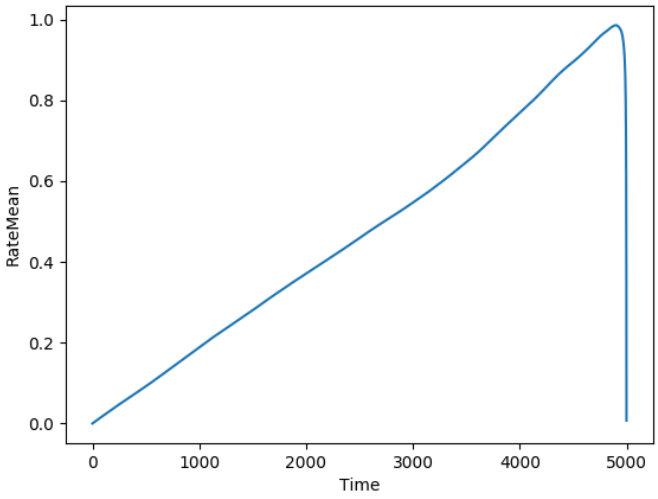
Lambda = 5


Jobs in queue
Queue 1
Queue 2
Time
0
1000
2000
3000
4000
5000


RateMean
Time
0.4
0.3
0.2
0.1
0.0
−0.1
0
1000
2000
3000
4000
5000

Lambda = 9


Jobs in queue
Queue 1
1500
1000
500
0
0
1000
2000
3000
4000
5000
Queue 2
30
20
10
0
Time


RateMean
1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000
Time

2.11 Conclusion

- Queue 2 là một hệ thống M/M/1 không có hồi tiếp nên các Metrics thu được từ mô phỏng gần như đúng với Lý thuyết;
- Queue 1 là một hệ thống M/M/1 có hồi tiếp, dẫn đến kết quả của các Metric thu được có phần chênh lệch so với lý thuyết, nguyên nhân này là do arrival rate của Queue 1 bị thay đổi, không còn đảm bảo phân phối mũ như ban đầu;
- Transient Remove với timeslice là 1 (đơn vị thời gian), tương đối lớn so với khoảng thời gian cho 1 lần Job Generating ($\sim$ 10-14 lần / s); điều này làm hiệu quả của Transient Remove không được biểu hiện rõ ràng trong quá trình mô phỏng;
- Independent Replications cho kết quả tương đối chính xác, phù hợp với kết quả quan sát được từ quá trình mô phỏng

THANKS FOR YOUR ATTENTION