

ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 556E Digital Solutions for Smart Cities

Instructors:

Sema Fatma Oktuğ

Yusuf Yaslan

Final Take Home Project Report

June 7, 2018

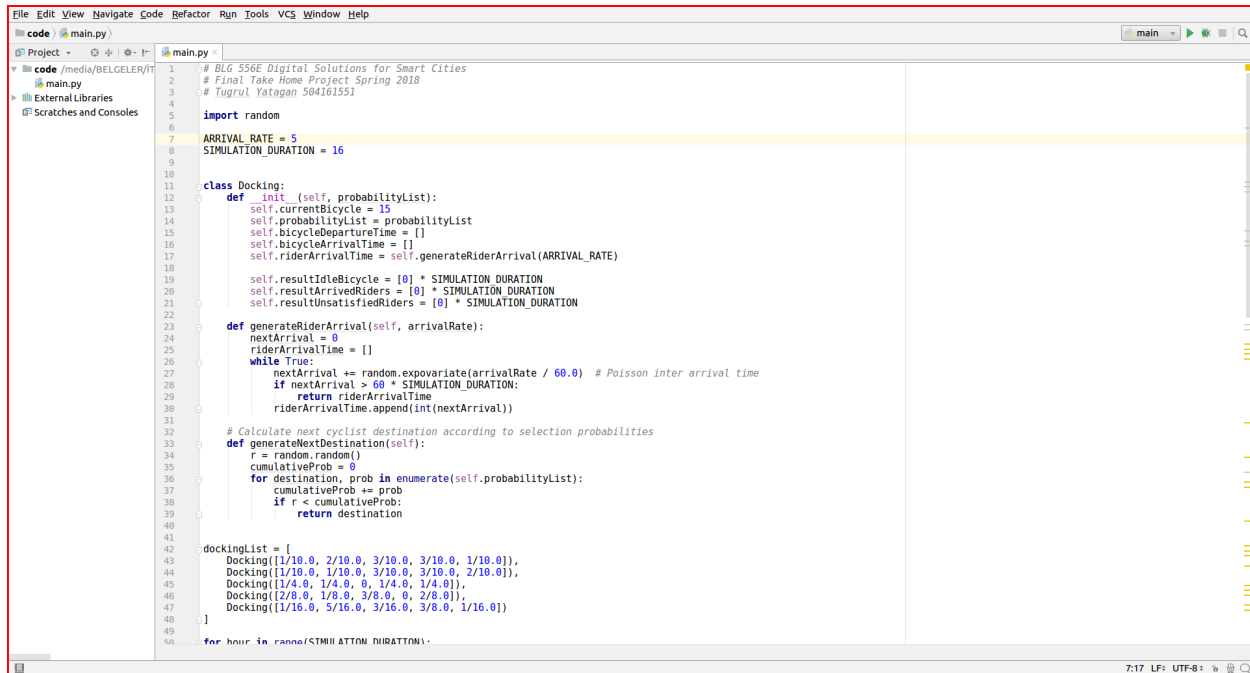
Tuğrul Yatağan

504161551

Development, Build and Test Environment

Ubuntu 16.04.4 LTS is used for build and test environment. Python 2.7 interpreter is used. PyCharm IDE is used for integrated development environment.

Example screen shot of development environment:



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
code /media/BELGELER/IT
Project - main.py
code /media/BELGELER/IT
main.py
External Libraries
Scratches and Consoles

1  # BLG 556E Digital Solutions for Smart Cities
2  # Final Take Home Project Spring 2018
3  # Tugrul Yatagan 584161551
4
5  import random
6
7  ARRIVAL_RATE = 5
8  SIMULATION_DURATION = 16
9
10
11 class Docking:
12     def __init__(self, probabilityList):
13         self.currentBicycle = 15
14         self.probabilityList = probabilityList
15         self.bicycleDepartureTime = []
16         self.bicycleArrivalTime = []
17         self.riderArrivalTime = self.generateRiderArrival(ARRIVAL_RATE)
18
19         self.resultIdleBicycle = [0] * SIMULATION_DURATION
20         self.resultArrivedRiders = [0] * SIMULATION_DURATION
21         self.resultUnsatisfiedRiders = [0] * SIMULATION_DURATION
22
23     def generateRiderArrival(self, arrivalRate):
24         nextArrival = 0
25         riderArrivalTime = []
26         while True:
27             nextArrival += random.expovariate(arrivalRate / 60.0) # Poisson inter arrival time
28             if nextArrival > 60 * SIMULATION_DURATION:
29                 return riderArrivalTime
30             riderArrivalTime.append(int(nextArrival))
31
32     # Calculate next cyclist destination according to selection probabilities
33     def generateNextDestination(self):
34         r = random.random()
35         cumulativeProb = 0
36         for destination, prob in enumerate(self.probabilityList):
37             cumulativeProb += prob
38             if r < cumulativeProb:
39                 return destination
40
41
42 dockingList = [
43     Docking([1/10.0, 2/10.0, 3/10.0, 3/10.0, 1/10.0]),
44     Docking([1/10.0, 1/10.0, 3/10.0, 3/10.0, 2/10.0]),
45     Docking([1/4.0, 1/4.0, 0, 1/4.0, 1/4.0]),
46     Docking([2/8.0, 1/8.0, 3/8.0, 0, 2/8.0]),
47     Docking([1/16.0, 5/16.0, 3/16.0, 3/8.0, 1/16.0])
48 ]
49
50 for hour in range(SIMULATION_DURATION):
```

Answers

Part a)

A discrete event base simulation is implemented for given problem. In this simulation; docking stations are implemented as objects which has their own current bicycle counter, cyclist departure time list, cyclist arrival time list and rider arrival time list. Rider arrivals are generated according to Poisson distribution. Every docking stations keeps selection probability list. They pop and push to target docking stations departure time list, cyclist arrival time list according to selection probability for target station.

Simulation results shows that bicycles are tends to accumulate into station 4 at the end of the day. It also clearly shows that increasing λ , increases unsatisfied rider ratio and decreases overall service quality of the bicycle sharing systems as expected.

Arrival rate $\lambda = 3$

To execute simulation for 3 riders per hour;

```
python blg556e_final_a.py 3
```

Following tables are result tables of simulation for $\lambda = 3$

Station 1	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	11	9	7	2	0	3	4	7	4	6	6	5	4	4	0
Number of riders arriving within that time slot	4	2	3	6	7	0	3	3	5	2	3	3	4	2	4	2
Number of unsatisfied riders	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2

Station 2	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	11	8	14	13	10	12	12	11	9	9	8	9	8	9	9
Number of riders arriving within that time slot	4	4	0	2	6	2	4	5	4	1	2	2	2	2	3	4
Number of unsatisfied riders	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Station 3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	13	16	15	12	14	12	11	11	11	12	16	16	15	15	15
Number of riders arriving within that time slot	2	3	3	6	1	4	4	2	2	3	0	4	6	2	4	4
Number of unsatisfied riders	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Station 4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	13	10	11	11	17	22	19	20	25	22	19	16	19	23	24
Number of riders arriving within that time slot	2	7	3	3	3	2	4	2	1	7	5	3	1	3	1	2
Number of unsatisfied riders	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Station 5	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	12	14	17	16	15	14	11	11	11	12	15	15	12	12	9
Number of riders arriving within that time slot	3	2	2	4	3	4	3	3	3	1	1	2	4	3	6	2
Number of unsatisfied riders	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For $\lambda = 3$;

$SQDS_i = [0.9434, 1.0, 1.0, 1.0, 1.0]$

$OSQBSS = 0.9878$

Arrival rate $\lambda = 5$

To execute simulation for 5 riders per hour;

```
python blg556e_final_a.py 5
```

Following tables are result tables of simulation for $\lambda = 5$

Station 1	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	8	9	7	8	6	0	2	2	8	12	13	16	20	19	24
Number of riders arriving within that time slot	7	4	7	5	6	9	3	2	2	1	3	2	4	3	1	8
Number of unsatisfied riders	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0

Station 2	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	8	0	3	4	5	4	9	10	6	0	2	2	2	0	2
Number of riders arriving within that time slot	7	13	2	6	5	6	3	3	8	11	2	3	7	4	3	4
Number of unsatisfied riders	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0

Station 3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	4	12	13	16	17	27	26	23	24	23	27	23	25	23	28
Number of riders arriving within that time slot	11	2	6	5	7	3	4	5	5	5	3	9	4	8	1	4
Number of unsatisfied riders	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Station 4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	9	11	11	8	8	7	14	13	13	14	7	0	3	1	0
Number of riders arriving within that time slot	6	8	8	5	4	4	3	7	2	6	11	8	4	7	8	4
Number of unsatisfied riders	0	0	0	0	0	0	0	0	0	0	0	0	2	0	4	2

Station 5	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Number of idle bicycles at the beginning of the time slot	15	10	13	15	14	13	9	7	5	0	2	4	4	6	6	7
Number of riders arriving within that time slot	5	3	4	4	4	8	6	5	11	4	3	8	2	4	6	4
Number of unsatisfied riders	0	0	0	0	0	0	0	0	4	2	0	0	0	0	0	0

For $\lambda = 5$;

SQDS_i = [0.9403, 0.9655, 1.0, 0.9158, 0.9259]

OSQBSS = 0.9490

Part b)

Selection probabilities of docking stations can be modeled as transition probabilities of a Markov chain. If we calculate steady state probabilities for this Markov chain, we will obtain the probability for a particular one of the states will approach a limiting value as time goes to infinity.

Python NumPy library is used for calculating steady state probabilities for each docking station;

Steady state probabilities = [0.1623, 0.1949, 0.2284, 0.2324, 0.1818]

If we multiply these probabilities with total number of bicycles. We will get steady state bicycle distribution;

Steady state bicycle distribution = [12.1739, 14.6209, 17.1335, 17.4354, 13.6360]

This distribution result is also matches well with simulation result from Part a, bicycles are expected to accumulate more at station 4 according to steady state distribution.

We need to come up with a solution to balance steady state bicycle numbers at docking stations. We can do that by distributing bicycles to docking station by inverse ratio of steady state probabilities. For example, we can decrease initial bicycle number at docking station 4 to balance bicycle numbers since bicycles are expected to accumulate at station 4 more anyway.

If we inverse these steady state probabilities;

Inverse of steady state probabilities = [0.2418, 0.2014, 0.1718, 0.1688, 0.2159]

If we multiply and round these inverse steady state probabilities with total number of bicycles. We will get inverse steady state bicycle distribution;

Optimal initial bicycle distribution = [18, 15, 13, 13, 16]

Simulation result shows that overall service quality of the bicycle sharing system is increases with these optimal initial bicycle distribution.

Arrival rate $\lambda = 3$

To execute simulation for 3 riders per hour with optimal initial bicycle distribution;

```
python blg556e_final_b.py 3
```

SQDS_i = [1.0, 1.0, 0.9787, 1.0, 0.9773]

OSQBSS = 0.9904

Arrival rate $\lambda = 5$

To execute simulation for 5 riders per hour with optimal initial bicycle distribution;

```
python blg556e_final_b.py 5
```

SQDS_i = [0.9242, 1.0, 1.0, 1.0, 0.9733]

OSQBSS = 0.9812

Part c)

An example real world case can be; most of the cyclist will travel station A to station B at morning and they will travel back from station B to station A at evening. So selection probabilities for station A and B will be swapped during the day. Also we should consider morning and evening rush hours. We can create a framework to observe user patterns then this framework can create a model for this usage patterns so that it can predict future request. With the help of this prediction scheme, bicycles can be distributed to stations accordingly. This can be achieved by machine learning or deep reinforcement learning methods.