COMP9334 Project Report

z5190669 Zheyuan Xu

In this project, a simulation program has been implemented. For the design problem, the smallest mean response time has been found as 0.8227 with given parameter values when fogTimeLimit is around 0.107 Statistically sound methods have been implemented and reproducibility has also been tested.

1.Probility distribution

1. inter-arrival time

The graph below shows the exponential distribution inter-arrival time with a mean arrival rate of 9.72.

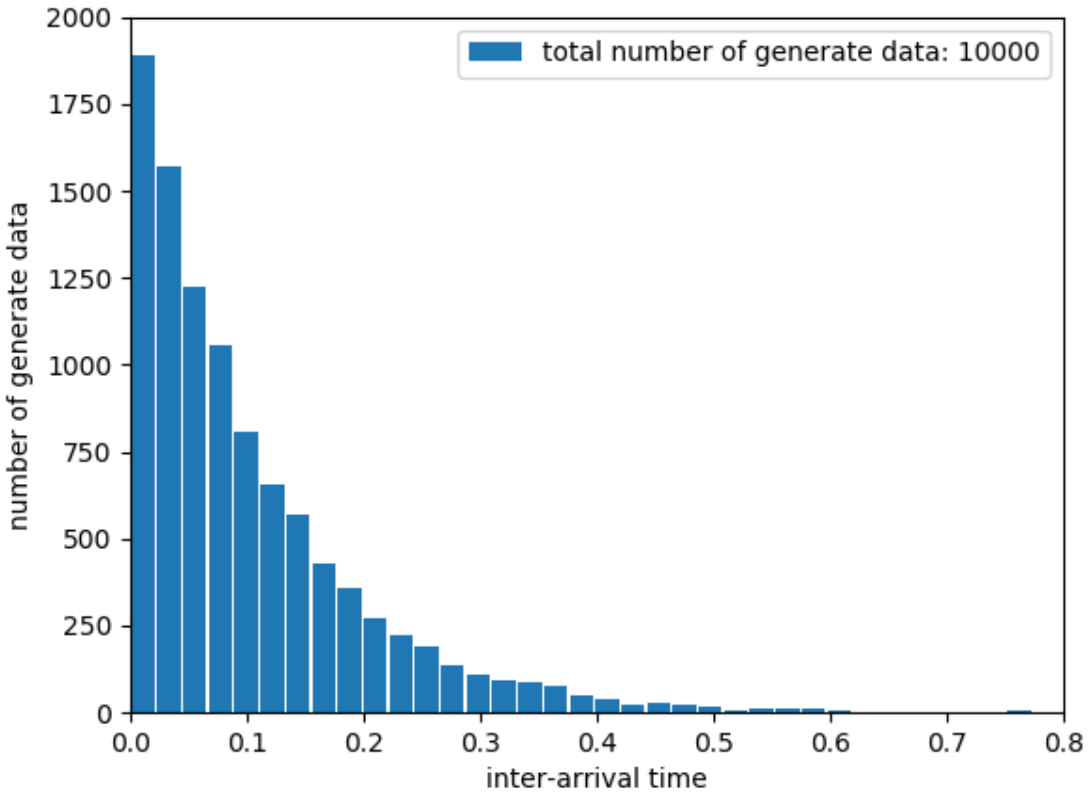


Fig1.distribution of inter-arrival time

In the program, *-np.log(1-np.random.uniform(0,1))/lambd* is used to generate inter-arrival time. As shown in Fig1, a total number of 10,000 inter-arrival time has been generated with a proper exponential distribution. (simulation code for this part is provided in draw.py)

2. service time

The graph below shows the exponential distribution service time with parameters given in section 5.2 where α1 = 0.01, α2 = 0.4, β = 0.86

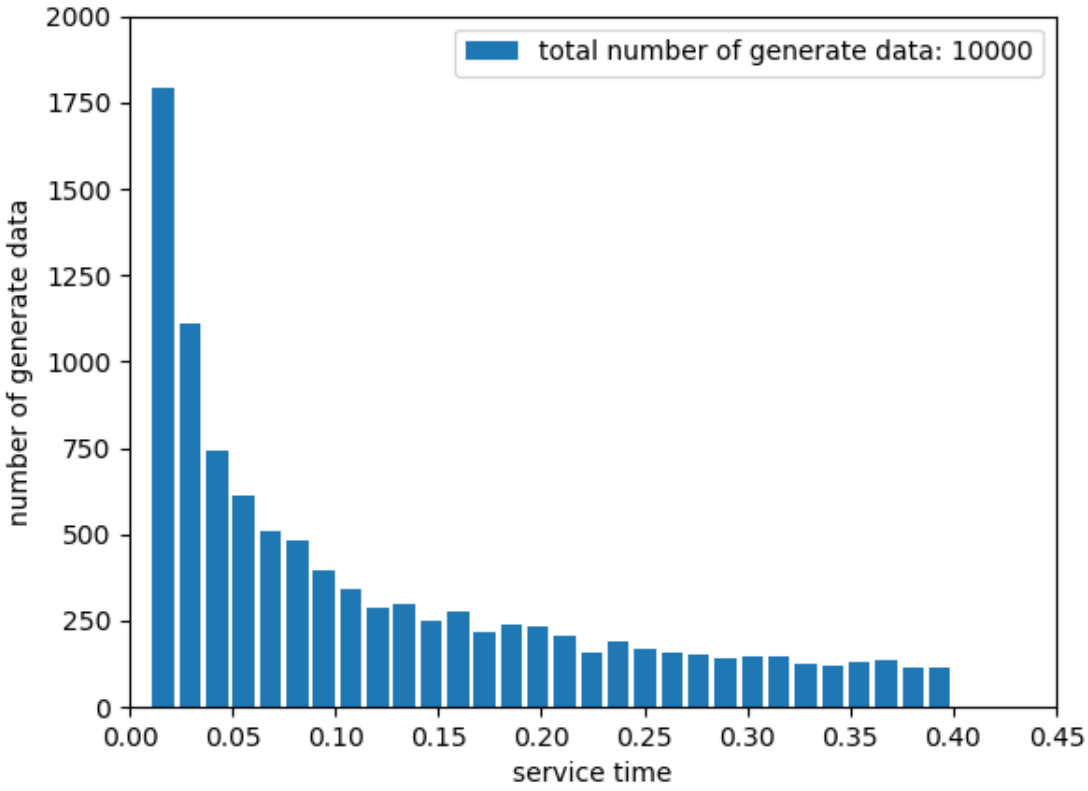


Fig2.distribution of service time

In the program, *(1-beta)/((alpha2**(1-beta))-(alpha1**(1-beta)))* is used to calculate the value of γ. The mathematical formula is provided in section 5.1.1 where $\gamma = \frac{1-\beta}{\alpha_2^{1-\beta} - \alpha_1^{1-\beta}}$.

np.random.uniform(0,1)(1-beta)/gamma + alpha1**(1-beta))**(1/(1-*

beta) is used to generate service time. As shown in Fig2, a total number of 10,000 service time has been generated with a proper exponential distribution according to the given parameter values. (simulation code for this part is provided in draw.py)

3. network latency

The graph below shows the uniformly distribution network latency in the open interval (v_1, v_2) with parameters given in section 5.2 where v_1 = 1.2, v_2 = 1.47

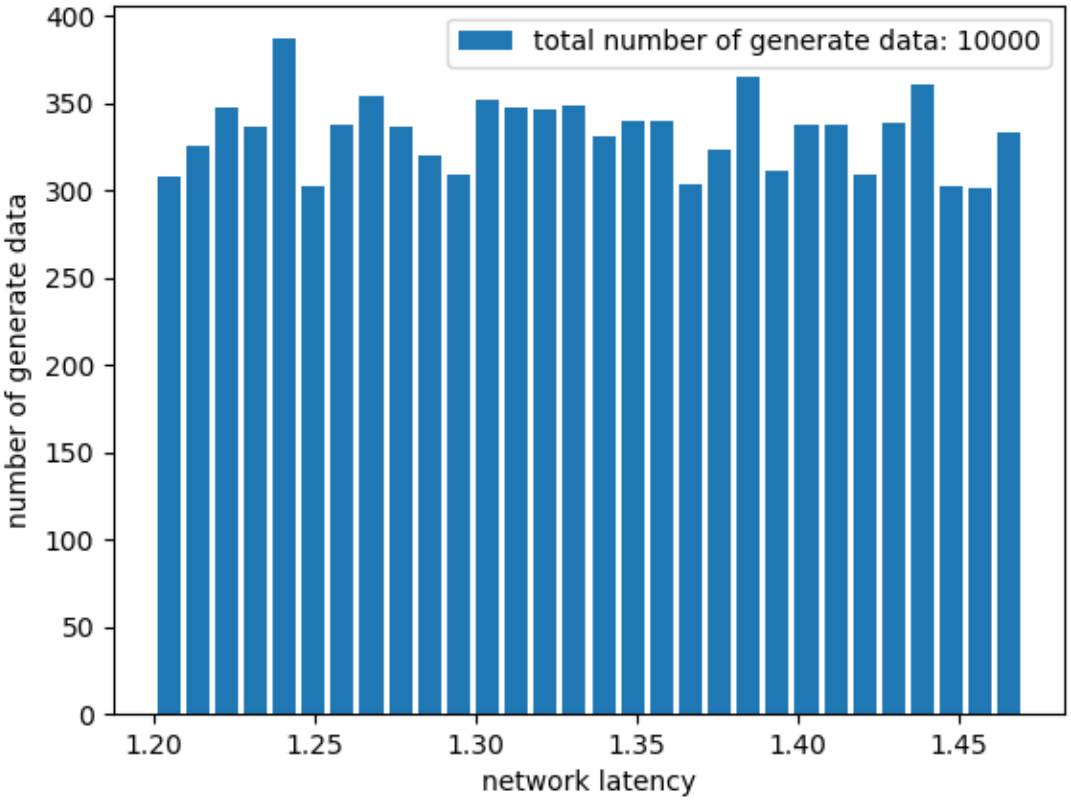


Fig3. distribution of network latency

In the program, *np.random.uniform(v1,v2)* is used to generate network latency. As shown in Fig3, a total number of 10,000 network latency time has been generated with a proper uniformly distribution between the open

interval (v_1, v_2). (simulation code for this part is provided in draw.py)

2. Verify trace simulation

Arrival time at the fog	Service time in the fog time unit	network latency
1	3.7	1.5
2	5.1	1.4
4	1.3	0
5	2.4	0
6	4.5	1.6

fogTimeLimit = 2.5, fogTimeToCloud = 0.7 which corresponding to the given project sample files: arrival_2.txt, service_2.txt, network_2.txt and para_2.txt.

joblist in fog:

```
masterclock:1.0 joblist:{1.0: 2.5}
masterclock:2.0 joblist:{1.0: 1.5, 2.0: 2.5}
masterclock:4.0 joblist:{1.0: 0.5, 2.0: 1.5, 4.0: 1.3}
masterclock:5.0 joblist:{1.0: 0.1666666666666669, 2.0: 1.1666666666666667, 4.0: 0.9666666666666668, 5.0: 2.4}
masterclock:5.666666666666667 joblist:{2.0: 1.0, 4.0: 0.8, 5.0: 2.2333333333333334}
masterclock:6.0 joblist:{2.0: 0.888888888888889, 4.0: 0.688888888888889, 5.0: 2.1222222222222222, 6.0: 2.5}
masterclock:8.755555555555556 joblist:{2.0: 0.19999999999999984, 5.0: 1.4333333333333331, 6.0: 1.811111111111111}
masterclock:9.355555555555556 joblist:{5.0: 1.2333333333333332, 6.0: 1.611111111111111}
masterclock:11.822222222222223 joblist:{6.0: 0.37777777777777755}
masterclock:12.200000000000001 joblist:{}
```

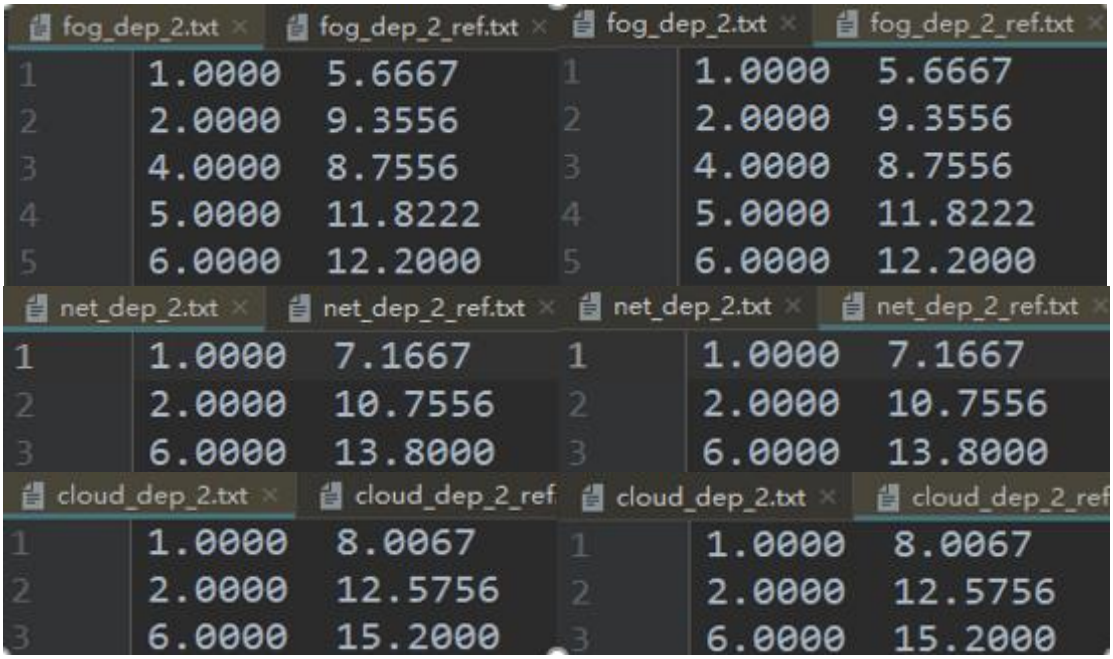
joblist in cloud:

```
masterclock:7.166666666666667 joblist:{7.166666666666667: 0.8400000000000001}
masterclock:8.006666666666668 joblist:{}
masterclock:10.755555555555556 joblist:{10.755555555555556: 1.8199999999999996}
masterclock:12.575555555555557 joblist:{}
masterclock:13.8 joblist:{13.8: 1.4}
masterclock:15.200000000000001 joblist:{}
```

The print out information looks exactly the same to the table provided in section 4.3

As can see from below, the simulation results are exactly the same to the data sample file provided. In addition, other simulation results are also identical to the sample files, which can illustrate that my simulation

program is correct.



Time	Request	Event
1	1	Arrival at the fog
2	2	Arrival at the fog
4	3	Arrival at the fog
5	4	Arrival at the fog
5.6667	1	Departure from the fog to the network
6	5	Arrival at the fog
7.1667	1	Departure from the network to the cloud
8.0067	1	Departure from the cloud
8.7556	3	Departure from the fog
9.3556	2	Departure from the fog to the network
10.7556	2	Departure from the network to the cloud
11.8222	4	Departure from the fog
12.2	5	Departure from the fog to the network
12.5756	2	Departure from the cloud
13.8	5	Departure from the network to the cloud
15.2	5	Departure from the cloud

(you can run wrapper.py to see the same result.)

table in section 4.3

3. Reproducibility

To proof the simulation result is reproducible, I used parameter values provided in section 5.2 and make fogTimeLimit = 0.11, time end = 1000,

seed = 32767, I kept these parameter values unchanged and repeat the simulation process for 1,000 times. The graph below shows the simulation result.

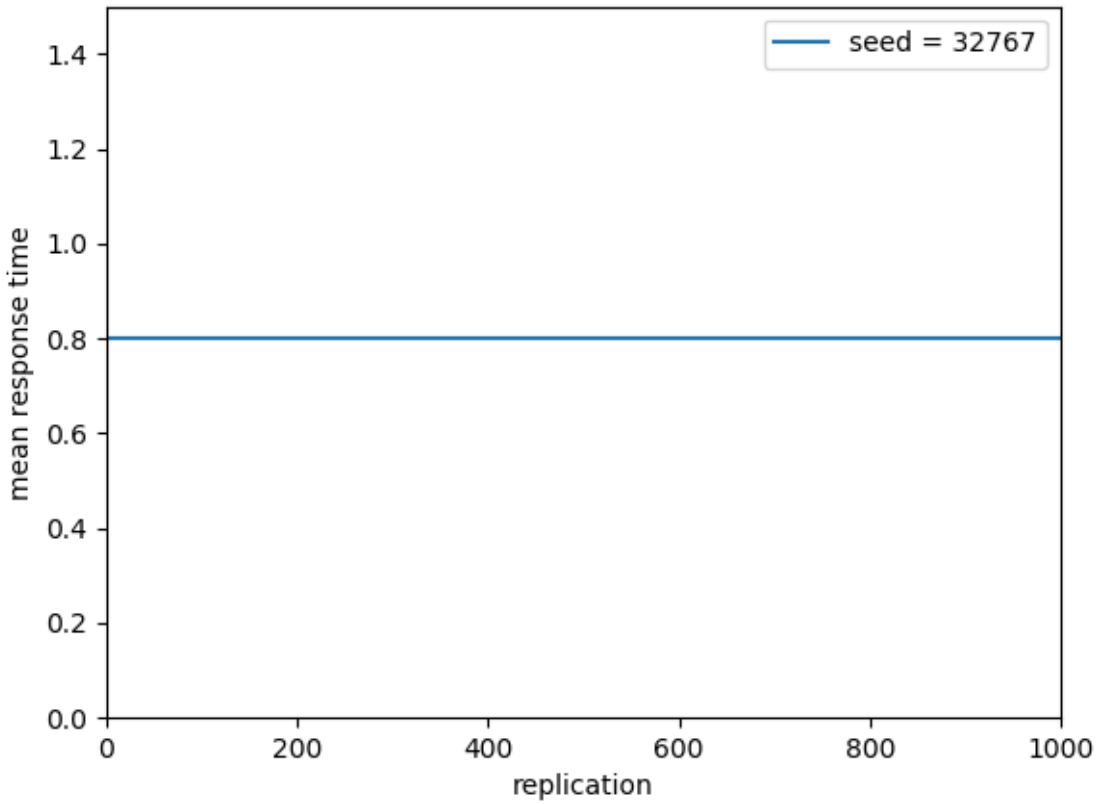


Fig4. Test for reproducible

The simulation result remains unchanged, proving that my simulation result is reproducible. (simulation code in same_seed.py)

4. Transient removal

To get transient end time, I used parameter values provided in section 5.2 and make fogTimeLimit = 0.11, end time loop from 1 to 1000 (step = 1), so there is total 1,000 simulations. The graph below shows the simulation result.

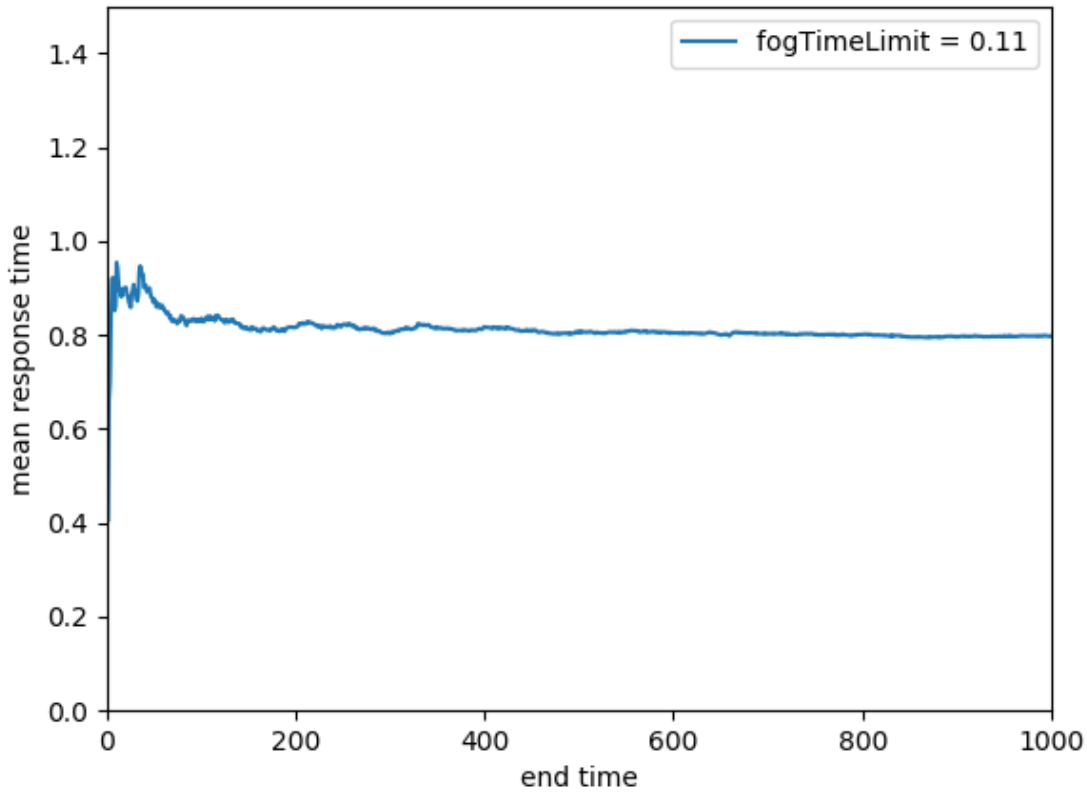


Fig5. find transient end time

By visual inspection, the transient end time is around 500, after this time, the mean response time curve becomes smooth enough, so when calculate the mean response time, we should make sure that end time is larger than 500 to get accurate value. In this project, I set end time to 1,000, this is long enough to avoid the imprecision caused by transient part. (simulation code in diff_end_time.py)

5. Determine the range of fogTimeLimit

To determine the range of fogTimeLimit and get the best mean response time for the system, I set end time = 1,000, seed = 32767 and loop fogTimeLimit from 0 to 0.2 (step = 0.01) according to the service time

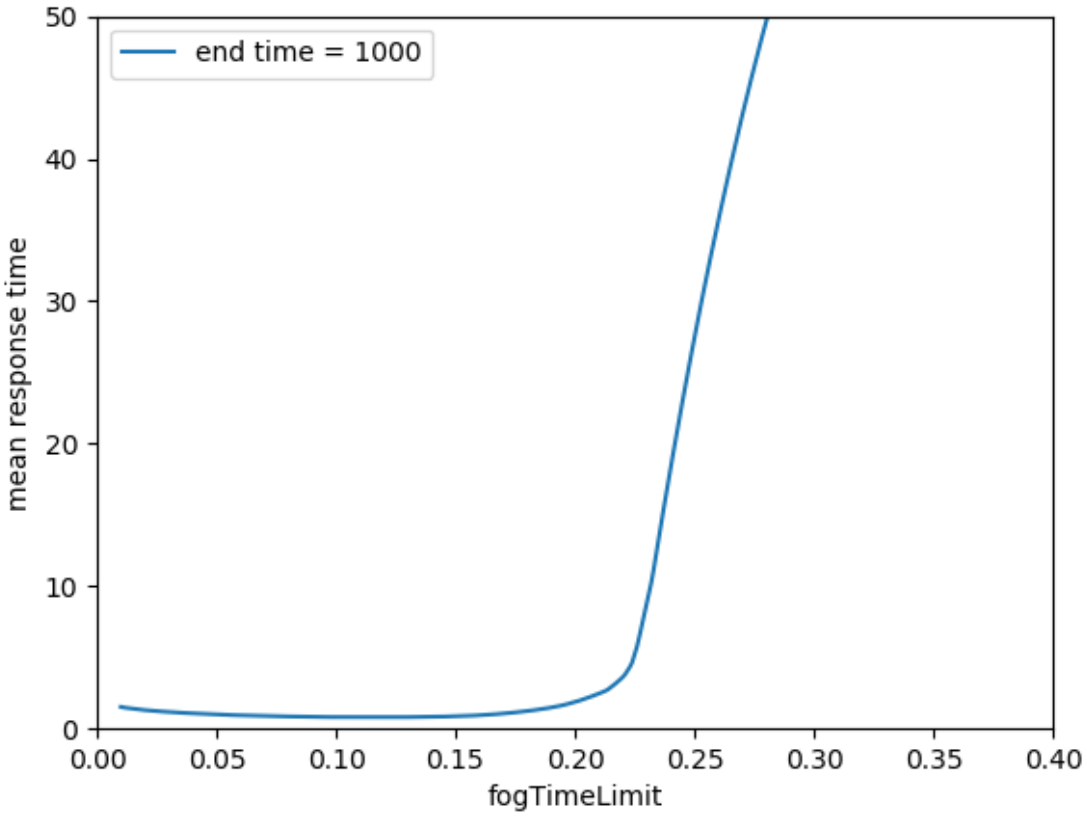


Fig6. find fogTimeLimit

As shown in Fig6, the best mean response time appeared in interval (0.08,0.15). In order to better observe the interval, another simulation has been implemented. Fig7 shows the second simulation result for this task. The best response time appeared in interval (0.106, 0.112), however, this is only the simulation when seed is unchanged, to get more precise result, we still need to compare results generate by different seeds, so I used 1,000 seeds for each fogTimeLimit in (0.11, 0.113) and the table below is the final mean response time I got:

fogTimeLimit	mean response time
0.106	0.822849
0.107	0.822705
0.108	0.822712
0.109	0.822788
0.11	0.823065
0.111	0.823421
0.112	0.823981

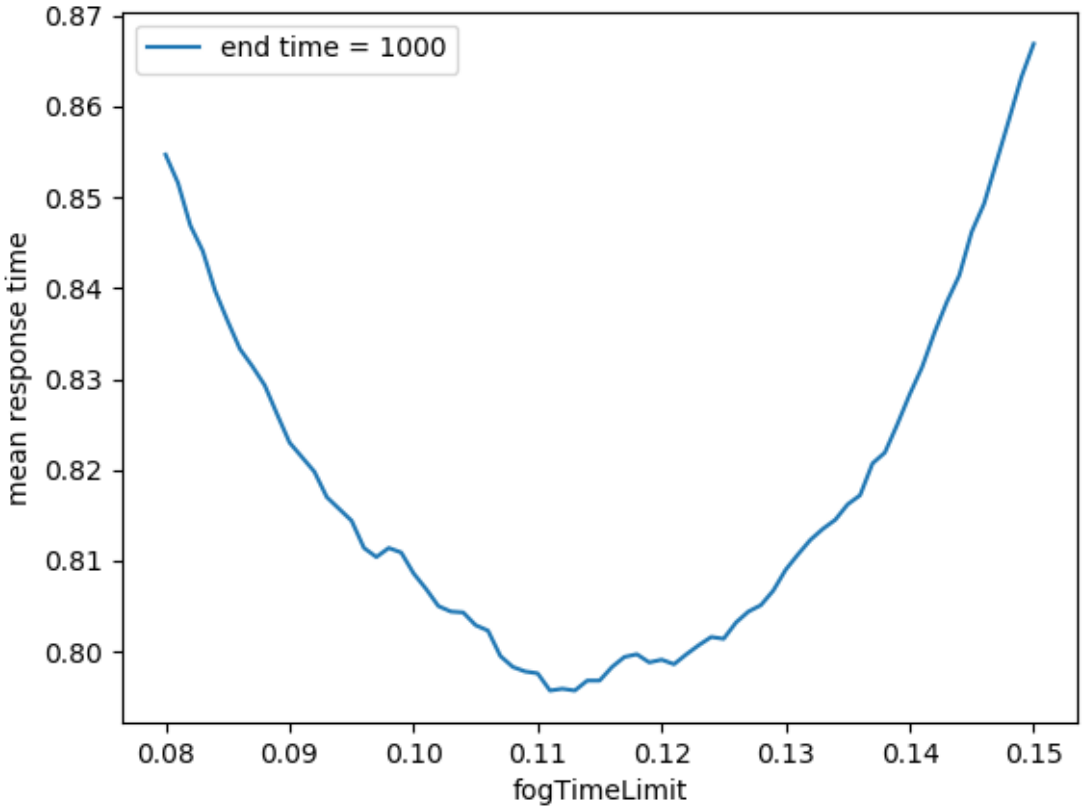


Fig7.find the best response time and fogTimeLimit

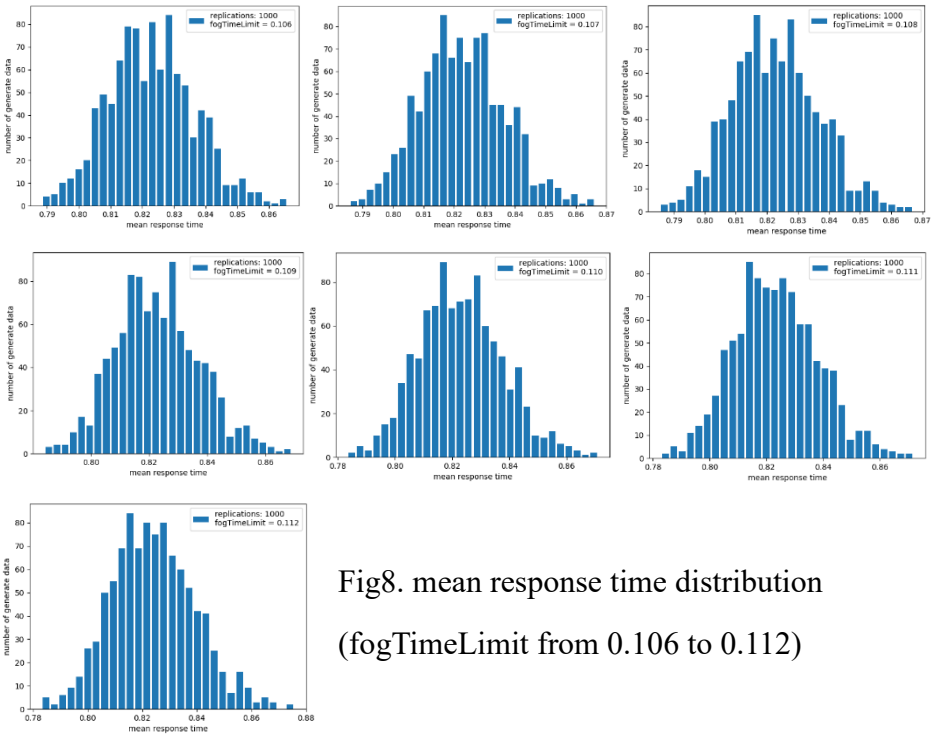


Fig8. mean response time distribution

(fogTimeLimit from 0.106 to 0.112)

the 95% confidence interval, $t_{1000,0.975} \approx 1.98$:

fogTimeLimit :0.106 response time: [0.8220049, 0.8236930]

fogTimeLimit: 0.107 response time: [0.8218409, 0.8235690]

fogTimeLimit: 0.108 response time: [0.8218329, 0.8235910]

fogTimeLimit: 0.109 response time: [0.8218907, 0.8236852]

fogTimeLimit: 0.110 response time: [0.8221489, 0.8239810]

fogTimeLimit: 0.111 response time: [0.8224874, 0.8243545]

fogTimeLimit: 0.112 response time: [0.8230286, 0.8249333]

As can see from above, the best confidence interval for response time appears between fogTimeLimit = 0.107 and fogTimeLimit = 0.108, however, when fogTimeLimit = 0.107, the interval range is relatively small. Thus, the best response time is around 0.8227 when fogTimeLimit is 0.107 (simulation code in sim_report.py and diff_seed.py)