MATH3840 Project 1

Police resource allocation / distribution

Case: Cops – Whatcha Gonna do?

Ross Hurley

Background Description

- The Case Study centers around a dilemma of allocating policeman and patrol cars to 5 different districts A, B, C, D, and E. The districts A-E currently have 2, 4, 5, 4 and 4 patrol cars respectively. The total number of police they currently have is unspecified.
- District D is encountering issues as residents are complaining that there aren’t enough police to cover the amount of crime occurring in the area.
- Some residents have complained that police are taking too long to arrive at the scene. Police are concerned that some of these crimes aren’t of a very high priority and the calls are very time consuming.

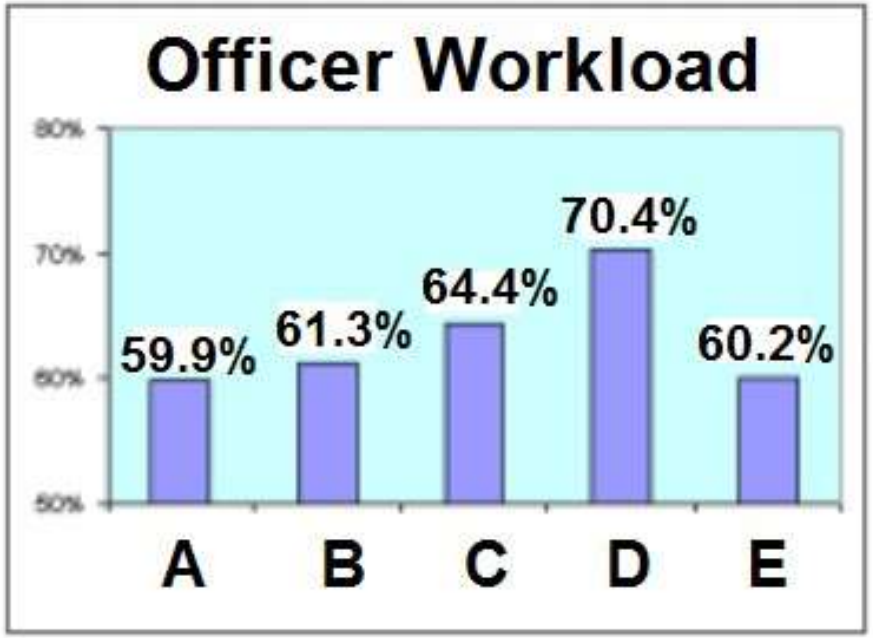
The situation

- We have 19 cars to distribute over 5 separate districts, each of which have varying levels of service demand.
- We are given the average call rate for each district as follows:
- District A : 1.9 calls per hour -> **ait of ~31.6** minutes
- District B: 4 calls per hour -> **ait of ~15** minutes
- District C: 4.75 calls per hour -> **ait of ~12.6** minutes
- District D: 4.15 calls per hour -> **ait of ~14.5** minutes
- District E: 3.75 calls per hour -> **ait of ~16** minutes

- **Where ait stand for average inter-arrival time i.e. 60 / calls per hour**

The situation

- The current allocation / distribution of cars is given as follows:
  2, 4, 5, 4, 4 for districts A – E respectively.
- We are given the findings from using this distribution:



District	Calls for Service (per hour)	Queueing Time (minutes)	Service Time (minutes)
A	1.9	21.1	37.8
B	4	7.2	36.8
C	4.75	6.7	40.7
D	4.15	14.9	40.7
E	3.75	7	38.5

The problem

- The question is – is this the best allocation? Well more correctly the questions are:
- What is the best allocation in terms of average queue waiting time (queue-ing times)?
- What is the best allocation in terms of server utilization (district workload)
- What is the best allocation if we wish to strike a balance between the two above criteria?

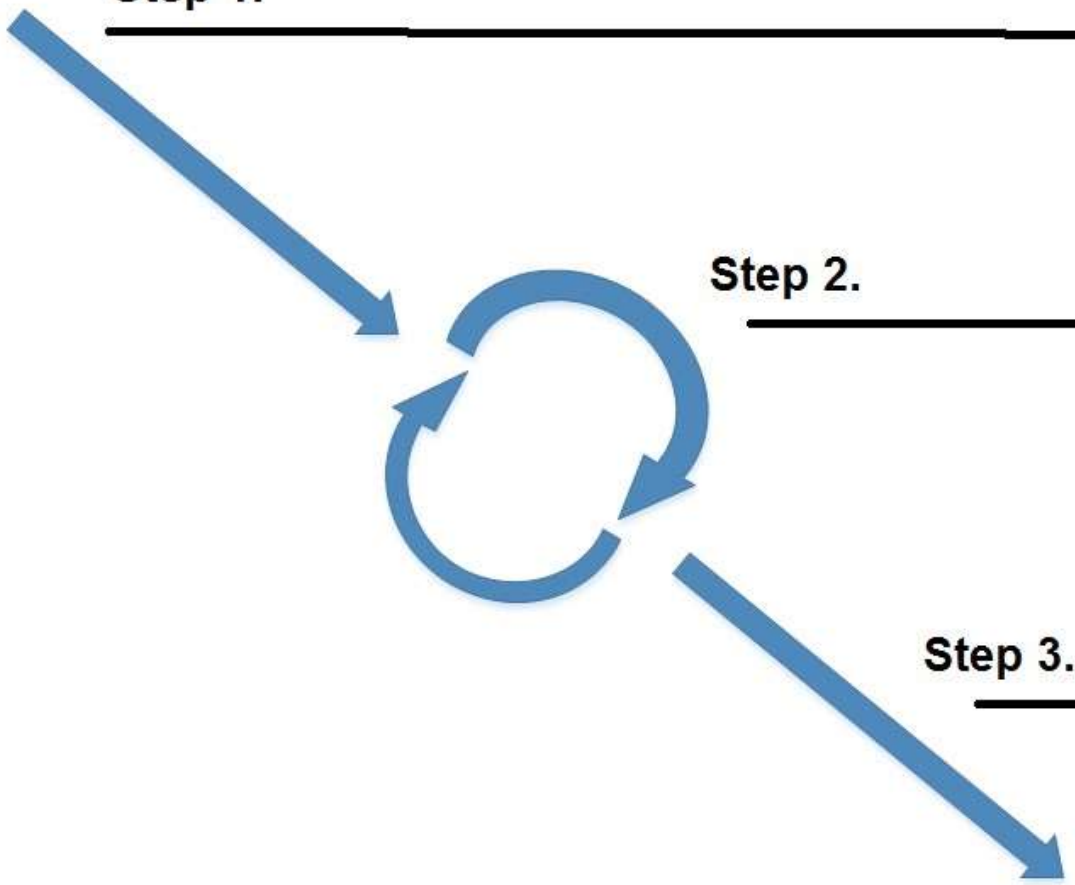
The model

- We mocked up a model from (more or less) scratch in the form of a discrete event simulation. Written in Javascript, run under node.js.
- It works essentially using a priority queue data structure. This priority queue can be thought of as the *time-line* of events.

Time	Events to process
0	Place call for District A
17	Return car for District B
23	Place call for District B
....	
100,000	Return car for District A
100,023	Return car for District D
100,050	Place call for District C

- At each iteration of step 2. (shown on next page) the top event of the time-line is *processed*. The process of an event leads to more events being added to this time-line. Say for instance the process of placing a call for district A will add an event of return car and place call for district A somewhere in the time-line.

The model - Simulation Anatomy



Step 1.

Initially place a call for each district, this 'seeds' the time-line

Step 2.

Process the timeline until the amount of events processed exceeds the limit given (50,000)

Step 3.

Compute and output relevant statistics

Key events for the simulation

- Placing a call:
  - Calculate inter-arrival time for the next call
  - If a car is available -> dispatch a car to this call (there is no wait time for calls when at-least one car is available), set that car as busy
  - If not -> add the call to waiting list

- Returning a car:
  - If a call is waiting -> dispatch a car to the first call on the waiting list (there is no leisure time between calls waiting)
  - If not -> set the car as ready for action

As each of the above events are processed from the time-line, relevant info is collected for statistical purposes: (total # of calls, total queue length etc)

Justifying Our Model

Running our simulation against the car distribution we showed earlier seems to quite closely match the given results for the current distribution.

```
>node simulation.js 500000 2 4 5 4 4

Reports:
District 0 average queue wait time: 21.376270931346617 || average car utilization: 61.4425
District 1 average queue wait time: 7.5783608324703025 || average car utilization: 63.24565
District 2 average queue wait time: 5.99857270554644455 || average car utilization: 65.65583999996005
District 3 average queue wait time: 15.020177917501345 || average car utilization: 72.36255
District 4 average queue wait time: 6.957171292314525 || average car utilization: 61.66085
```

Note that the precision of our simulation will only benefit from a greater and greater event limit. But at 500,000 the simulation seems to consistently stick to the expected results closely. We chose 500,000 events as it seemed to strike a good balance between result precision and simulation run-time.

Justifying Our Model (cont)

In the case file it can be calculated that their results were found using a total amount of calls: 5426, 11424, 13566, 11852, 10710 for Districts A – E respectively.

As can be seen in the figure below we have much more than enough just to ensure we are actually processing enough events (achieved with 500,000 event limit).

We believe that our model is robust enough to compute ***sane*** queue-ing times and car utilizations (server utilization) using a 500,000 event limit.

Our approach – attempted heuristic

- We firstly tried to use a simple heuristic in order to allocate car resources.
- This heuristic is essentially based on call for service and service resource.
- Using the average inter-arrival time (shown earlier), let us call that ***ait*** and also using the average service time, let us call that ***s***.
- We reasoned that:
  - if s ≤ ait, than the allocation for that district was adequate
  - If s > ait, then things get a little more complicated:
    - We would compute the allocation via **ceil(s / ait)**

Our approach – attempted heuristic

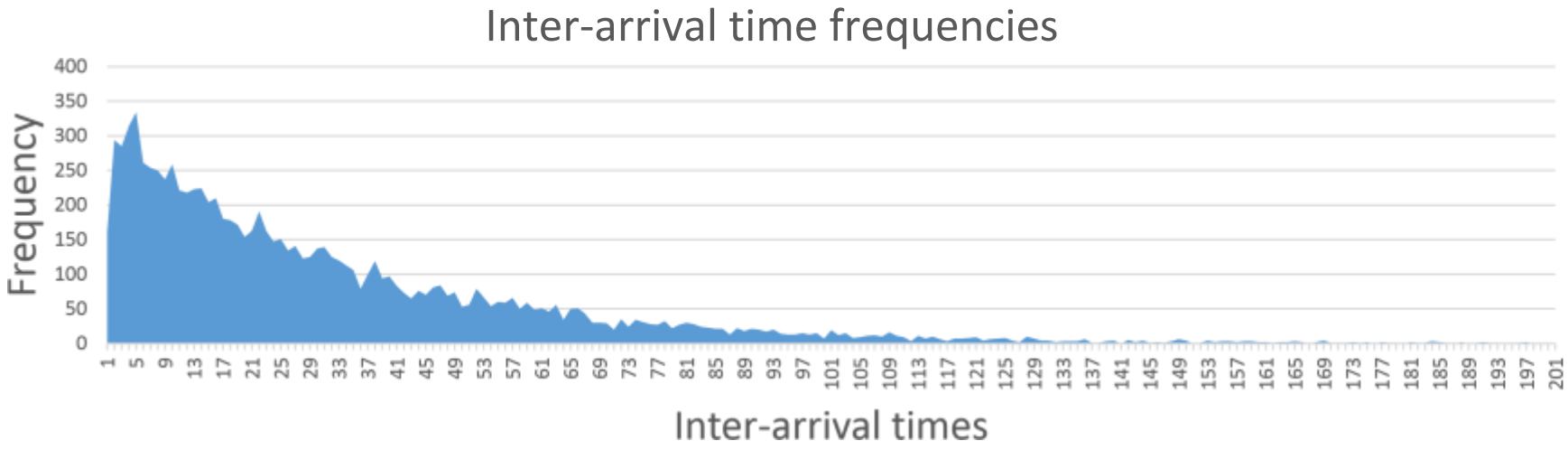
- For our districts this would compute to be 2, 3, 4, 3, 3. Which is actually suboptimal (2, 4, 5, 4, 4 does better in terms of queue-ing time). Does this mean we simply evenly distribute the other 4 of the cars?

- This heuristic does not take into account the random nature of the simulation nor does not it say anything about allocating with consideration to service utilization.

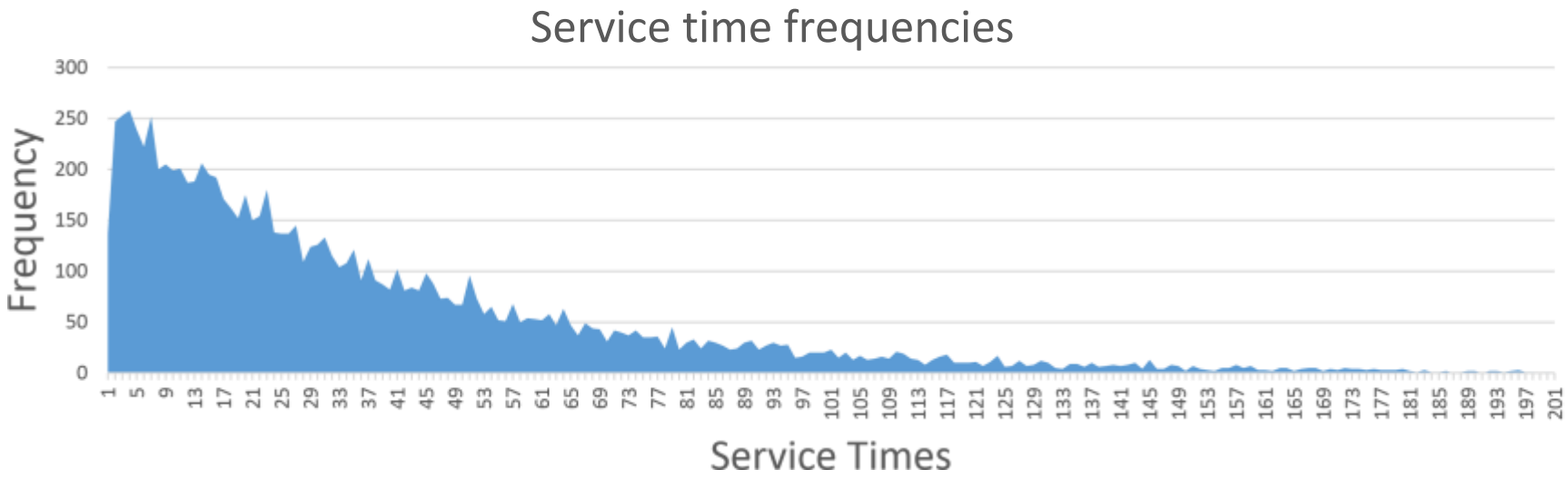
- ***How do # of cars, call rate and service times relate ??***

- ***Is there even any heuristic method that would minimize service utilization ??***

Our approach – attempted heuristic



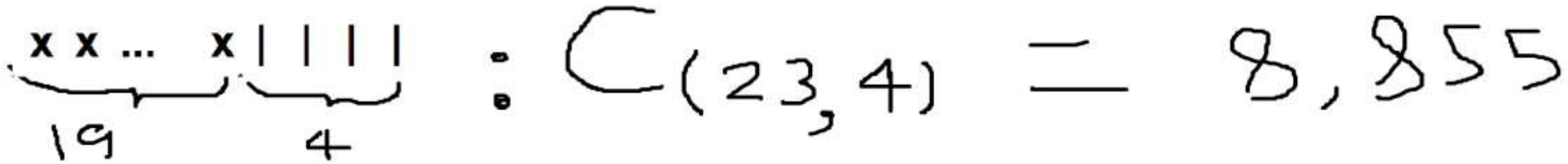
District A Inter-arrival times



District A Service times

New approach – distribution brute force

- There are 19 cars to distribute over 5 districts.
- How many different combinations of car distributions are there??
- This can actually be seen as a combinations with repetitions problem



- There are 8,855 different combinations – This amount is feasible to check given one simulation takes ~0.5 second (~1.25 hours of brute force).

Evaluating each car distribution

- Each car distribution is given three separate scores:
  - Queue time Score (Sum of District A – E average queue-ing times)
  - Service Utilization Score (Sum of District A – E average service utilization)
  - Queue time + Service Utilization score

- There are three 'best' car distributions kept:
  - Best distribution according to queue-ing criteria
  - Best distribution according to service utilization criteria
  - Best distribution according to a balance of the two above criteria

```
19, 0, 0, 0, 0
18, 1, 0, 0, 0
18, 0, 1, 0, 0
...
2, 4, 5, 4, 4
...
0, 0, 1, 0, 18
0, 0, 0, 1, 18
0, 0, 0, 0, 19
```

Brute force results (19 cars)

```
trying 0,0,1,0,18
trying 0,0,0,19,0
trying 0,0,0,18,1
trying 0,0,0,17,2
trying 0,0,0,16,3
trying 0,0,0,15,4
trying 0,0,0,14,5
trying 0,0,0,13,6
trying 0,0,0,12,7
trying 0,0,0,11,8
trying 0,0,0,10,9
trying 0,0,0,9,10
trying 0,0,0,8,11
trying 0,0,0,7,12
trying 0,0,0,6,13
trying 0,0,0,5,14
trying 0,0,0,4,15
trying 0,0,0,3,16
trying 0,0,0,2,17
trying 0,0,0,1,18
trying 0,0,0,0,19
best config according to queue time: 2,4,5,4,4
best config according to workload: 3,5,1,5,5
bets config according to both: 3,4,4,4,4
```

Brute force results / Recommendations (19 cars)

Best of queue-ing results

```
>node simulation.js 500000 2 4 5 4 4

Reports:
District 0 average queue wait time: 21.376270931346617 || average car utilization: 61.4425
District 1 average queue wait time: 7.5783608324703025 || average car utilization: 63.24565
District 2 average queue wait time: 5.9985727055464455 || average car utilization: 65.65583999996005
District 3 average queue wait time: 15.020177917501345 || average car utilization: 72.36255
District 4 average queue wait time: 6.957171292314525 || average car utilization: 61.66085
```

Best of service utilization results

```
>node simulation.js 500000 3 5 1 5 5
Reports:
District 0 average queue wait time: 2.9737648104708634 || average car utilization: 40.800266666669603
District 1 average queue wait time: 1.8190294707095003 || average car utilization: 50.8291600000093
District 2 average queue wait time: 308198.3752297222 || average car utilization: 99.999
District 3 average queue wait time: 3.6577779506890193 || average car utilization: 58.330159999999082
District 4 average queue wait time: 1.5986332631291163 || average car utilization: 50.06412000000016814
```

Best of both results

```
>node simulation.js 500000 3 4 4 4 4
Reports:
District 0 average queue wait time: 2.7114538333366071 || average car utilization: 41.822200000003104
District 1 average queue wait time: 7.5703759088014815 || average car utilization: 63.69355
District 2 average queue wait time: 31.31251602045628 || average car utilization: 81.4497
District 3 average queue wait time: 13.768050393910215 || average car utilization: 71.9241
District 4 average queue wait time: 7.247879048774345 || average car utilization: 62.03515
```

Brute force results / Recommendations (19 cars)

- Best configuration according to queue-ing score provides no reduction in queue waiting times or service. Since it is was found to be our original configuration.

- Best configuration according to server utilization provides a heavy increase in queue waiting time (See District c). But for each district (**except District C where server utilization is increased by ~34%)** for the other districts it had a net reduction of **~59%.**

- Best configuration according to both criteria keeps all queue-ing times virtually the same except for **district A (reduced queue time by ~20 mins), and District C (increased by ~15 mins).** It also Keeps all server utilizations virtually the same (except for District C where it is increased by ~15%).

Brute force results (28 cars)

```
trying 0,0,0,21,7
trying 0,0,0,20,8
trying 0,0,0,19,9
trying 0,0,0,18,10
trying 0,0,0,17,11
trying 0,0,0,16,12
trying 0,0,0,15,13
trying 0,0,0,14,14
trying 0,0,0,13,15
trying 0,0,0,12,16
trying 0,0,0,11,17
trying 0,0,0,10,18
trying 0,0,0,9,19
trying 0,0,0,8,20
trying 0,0,0,7,21
trying 0,0,0,6,22
trying 0,0,0,5,23
trying 0,0,0,4,24
trying 0,0,0,3,25
trying 0,0,0,2,26
trying 0,0,0,1,27
trying 0,0,0,0,28
best config according to queue time: 4,5,7,6,6
best config according to workload: 4,6,6,6,6
bets config according to both: 4,6,6,6,6
```

We also consider the distribution of 28 cars (new recruit scenario).

# of distributions = 35,960
Still a feasible amount if each simulation takes ~0.5 seconds (~5 hours of brute force).

Brute force results / Recommendations (28 cars)

Best of queue-ing results

```
>node simulation.js 500000 4 5 7 6 6
Reports:
District 0 average queue wait time: 0.5221439848498718 || average car utilization: 31.3246
District 1 average queue wait time: 1.78915909552482l8 || average car utilization: 51.449720000001553
District 2 average queue wait time: 0.5572084602905951 || average car utilization: 47.702285714298526
District 3 average queue wait time: 1.054526714839357 6 || average car utilization: 48.4717666666643
District 4 average queue wait time: 0.4119605751074468 || average car utilization: 41.789333333333275
```

Best of service utilization and both results

```
>node simulation.js 500000 4 6 6 6 6
Reports:
District 0 average queue wait time: 0.535733169347138 || average car utilization: 31.44035
District 1 average queue wait time: 0.47573863698649566 || average car utilization: 42.2186999999965
District 2 average queue wait time: 1.76869468772838 53 || average car utilization: 55.377266666675
District 3 average queue wait time: 0.93595654359505 36 || average car utilization: 48.530866666669525
District 4 average queue wait time: 0.4292166105333248 || average car utilization: 41.443033333326945
```

Short comings of our model / approach

- When a simulation stops processing events (ie # of events processed exceeds limit) and there are still calls waiting in the system. These calls are accounted for but it becomes an *under-estimate.*

- Our brute force method will be an obvious bottleneck to find optimal solutions as the # of cars to distribute increases. In fact it grows by the order of O(n^4).

- Node.js will become a less and less suitable platform to run simulations with more and more resources (ie 1000's of cars and / or 1000's of districts) as Node.js is single-threaded and as such cannot take advantage of multi-core CPUs or multiple CPUs. Node.js was used here because of the capability to rapidly prototype and implement ideas.

- The metric '***best configuration according to both*** criteria' needs to be re-considered in the way that is calculated. Currently it is calculated via Queue time + Service Utilization. This is not a good balance as Service Utilization's maximum possible value is 100 where as the Queue time can be indefinitely large. A better way to calculate this would be to see firstly calculate the best Queue time, called ***Q'*** and then compute the score as (*1 –* ***Q'*** */ current queue time*). This way we know the range of possible values of this score ***[0, 1)*** where the best score, *1 –* ***Q'/Q' = 0*** and we know the increasingly suboptimal scores will be approaching 1 e.g. let ***current queue time = 1000*** and let ***Q' = 25*** then the score will be ***0.975.***