IE 709: Assignment 2 Report

Submitted by: Aakash Banik (16i190010) and Shivangi Saklani (16i190008)

1 Question 1

We did the entire first question under a single program **e**x1.py as we used the data generated from one subpart on the later subparts.

We divided the entire area into 50x50 small pixels of area $\frac{l}{k} * \frac{w}{k}$ each. As a result we came up with a coordinate matrix of order 51x51 whose $[i, j]^{th}$ element denotes the coordinate of the point of intersection of the $(i+1)^{th}$ horizontal line starting from above and the $(j+1)^{th}$ vertical line starting from left. Basically we assumed the top left coordinate is our origin i.e.,the [0,0] coordinate and the bottom right coordinate is the [51,51] coordinate. We named this matrix "coord".
Next we computed the co-ordinates of the mid-point of each of the pixels and stored each of them in the "pixels" matrix. So, pixels[i][j] stores the co-ordinates of the mid point of the pixel corresponding to the i^{th} row and the j^{th} column.

1.1 Assumptions:

- We assumed rectilinear distances between two locations in order to compute the distance between them.
- We assumed that the population is uniformly distributed inside a pixel and also across all pixels.
- We assumed that the speed of travel is constant across all pixels. For simplicity we considered the speed to be 1 unit/time units. Hence, the time required to reach some place A from B would basically be the rectilinear distance between them.
- No call would come to an EMS when its in use.

So in order to draw a map of areas served by each facility assuming that each call is served by the facility nearest to it, we computed for each pixel the distances of all N facilities from it and mark the pixel with the facility no. which is closest to it. We plotted this map(given below) to see the coverage of each EMS facility across the city.

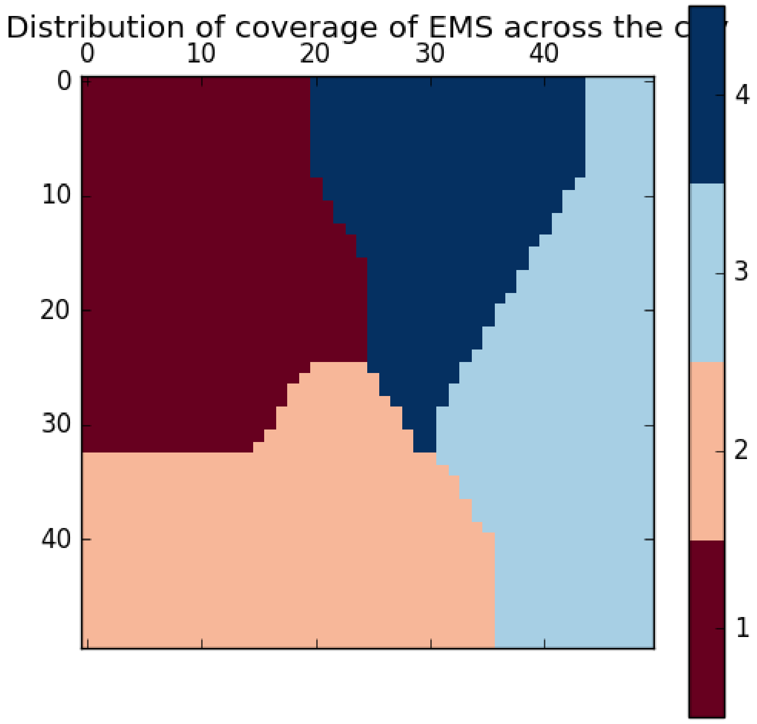
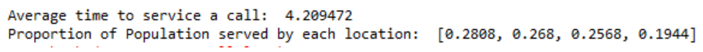


Figure 1: Coverage of each EMS across the city

The average time to service a call for the whole region is computed by summing over the times an EMS would take to reach the pixels which comes under its coverage and then dividing the entire sum by total no. of pixels.
Since we assumed that the population is uniformly distributed across the region, the proportion of population served by each EMS is basically the total no. of pixels that comes under its coverage divided by total no. of pixels.



```
Average time to service a call:  4.209472
Proportion of Population served by each location:  [0.2808, 0.268, 0.2568, 0.1944]
```

Figure 2: Average time to service a call and proportion of population served by each EMS

Now, since the proportion of population covered by each of the EMS are not equal, we need to balance out the load across all EMS-es by doing a re-allotment of the coverage of each EMS.
We do this by solving the following optimisation problem. It should be noted that by doing this, the avg. time to service a call is likely to increase a bit but the proportion of population covered by each EMS would be more or less equal.

Optimisation Problem

Let, x_{ij} be an indicator variable which takes the value 1 if the i^{th} EMS serves location j ; 0 , otherwise. i=1(1)N. $j = 1(1)k^2$.
Also, d_{ij} denotes the distance between the i^{th} EMS station and the j^{th} location. i=1(1)N. $j = 1(1)k^2$
N is the total no. of EMS stations.

The objective function is given as:
minimize: $\sum_{i=1}^{N}\sum_{j=1}^{k^2} x_{ij} * d_{ij}$

subject to the constraints:

$1 \leq \sum_{j=1}^{k^2} x_{ij} \leq [\frac{k^2}{N}] + 1 \ \forall i = 1(1)N$ [To make sure that the max. no. of pixels that an EMS would provide service to are roughly equal for all stations]

$\sum_{i=1}^{N} x_{ij} = 1 \ \forall j = 1(1)k^2$ [To make sure that each pixel is served by only 1 EMS]

The above optimisation problem gives the following plot showing the coverage of each EMS in the region.

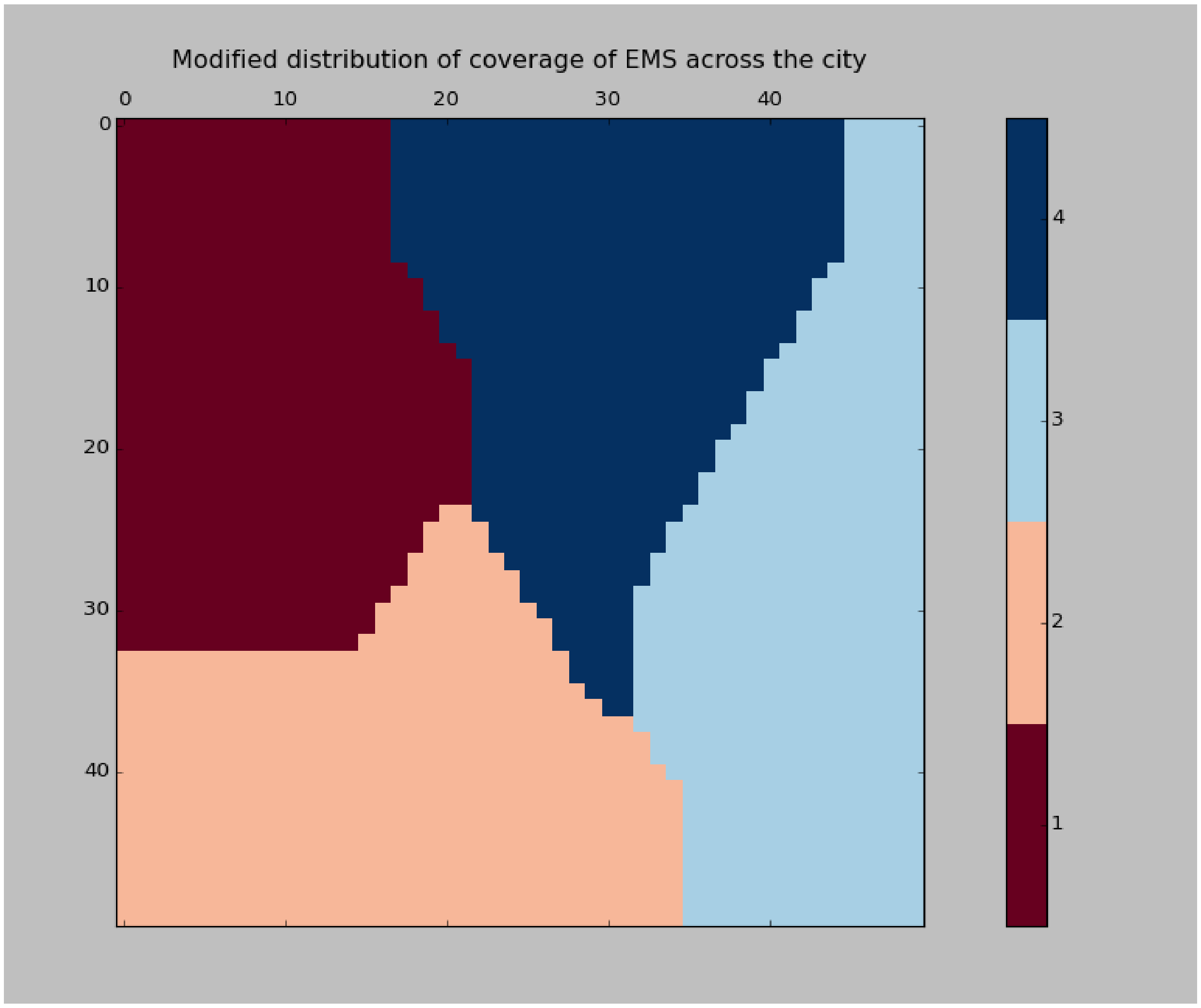
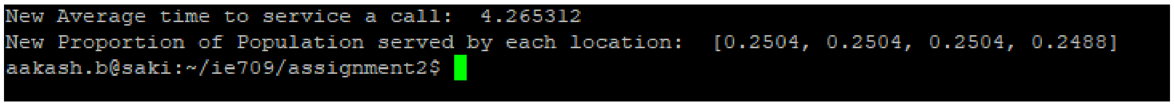


Figure 3: Modified distribution of coverage of each EMS in the region

Results show that this new distribution of pixels balances out the load on each EMS uniformly. It also shows that the average time to service a call increased to 4.265312 time units from 4.209472 time units.



```
New Average time to service a call:  4.265312
New Proportion of Population served by each location:  [0.2504, 0.2504, 0.2504, 0.2488]
aakash.b@saki:~/ie709/assignment2$
```

Figure 4: Average time to service a call and proportion of population served by each EMS after re-allocation of the pixels

2 Question 2

2.1 Assumptions:

- We assumed that both the doctors as well as the patients are punctual, i.e. they arrive exactly when the appointment is scheduled.
- We also assumed that the service time for each patient is Gamma distributed with mean 12 mins and variance 9.

2.2 Subpart a

We ran the simulation for 10^5 times. The code is attached with the name **ques21.py**.The histogram of waiting times is given below:

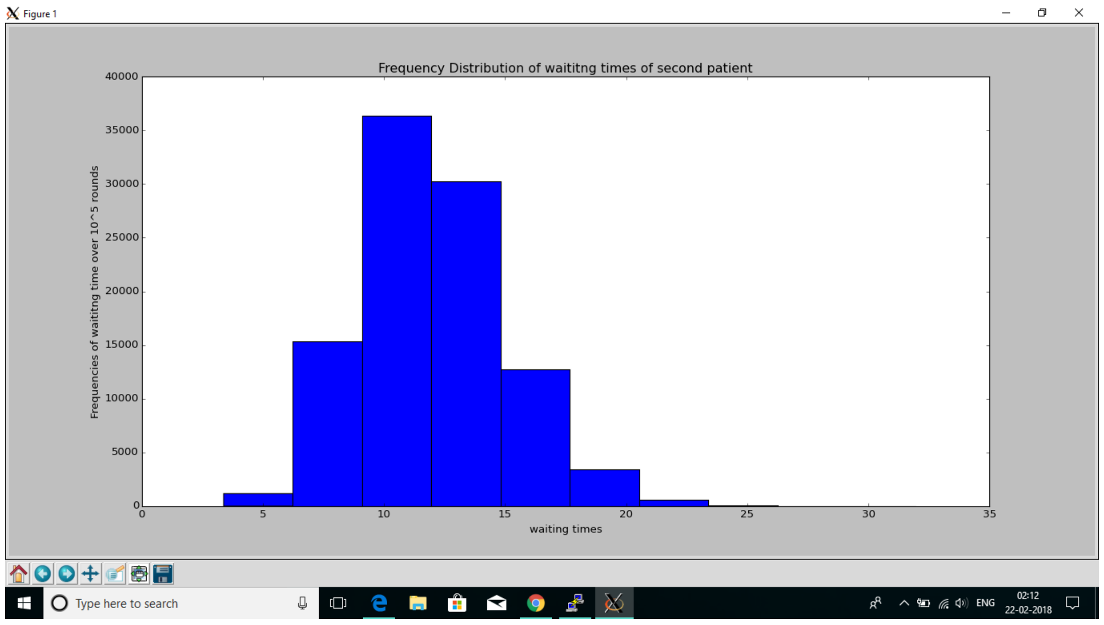
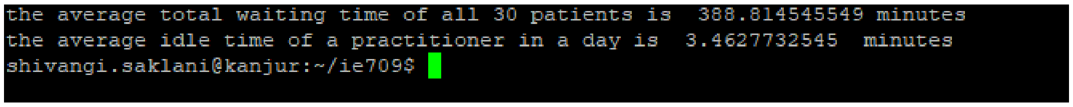


Figure 5: Histogram for Waiting Times of the second patient

We can see from the histogram above that the distribution of waiting times for the second patient is positively skewed. The mode of the distribution is around the 10.5 mins mark. Also the probability that the waiting time is less than 5 mins or greater than 20 mins is negligible. This distribution of waiting times is more or less a good fit to our setting if we assume that patients come only to get checked and not to report a follow-up(since the follow-ups take barely a minute or two; which is not picturised in our histogram).

2.3 Subpart b

We ran the simulation for 10^5 times. The code is attached with the name **ques22.py**.
The average sum of all waiting times of patients when Bailey and Welch's rule is followed is: 388.81454
The total idle time of the doctor when Bailey and Welch's rule is followed is: 3.46277

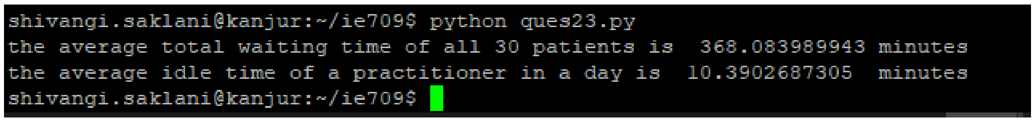


```
the average total waiting time of all 30 patients is  388.814545549 minutes
the average idle time of a practitioner in a day is  3.4627732545  minutes
shivangi.saklani@kanjur:~/ie709$
```

Figure 6: Avg waiting time of patients and doctor's idle time in Bailey and Welch's rule

2.4 Subpart c

We ran the simulation for 10^5 times. The code is attached with the name **ques23.py**.
The average sum of all waiting times of patients when Soriano's rule is followed is: 368.0839
The total idle time of the doctor when Soriano's rule is followed is: 10.3902



```
shivangi.saklani@kanjur:~/ie709$ python ques23.py
the average total waiting time of all 30 patients is  368.083989943 minutes
the average idle time of a practitioner in a day is  10.3902687305  minutes
shivangi.saklani@kanjur:~/ie709$
```

Figure 7: Avg waiting time of patients and doctor's idle time in Soriano's rule

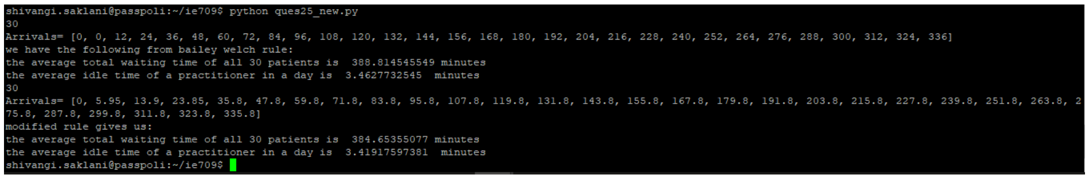
2.5 Subpart d

No, none of the above rules dominate the other. This is beacause Bailey and Welch's rule fares better in terms of Idle time of doctor while Soriano's rule fares better in terms of waiting time of patients.

2.6 Subpart e

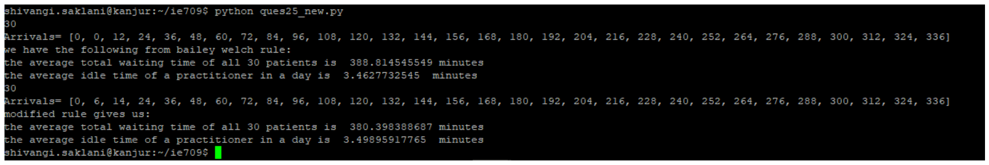
We ran the simulation for 10^5 times. The code is attached with the name **ques25.py**.

In this method, we start with increasing inter arrival times(IAT) and eventually make them constant. In this particular implementation, we have taken IATs starting from 5.95mins with an increment of 2mins. This is done for the first four arrivals and the rest of the arrivals have a constant IAT of 12 mins.
Unlike the Bailey and Welch's method, we have only one arrival at every time point. It is clear from the output that for above mentioned IAT pattern and single arrival at every time point, we can clearly conclude that our scheduling rule strictly dominates the Bailey and Welch Rule.
It might be an argument that IAT of 5.95 minutes is not practically feasible, but it can be seen that, even for IAT as 6minutes with increment of 2 minutes for first 4 arrivals, we are pretty close to the Bailey-Welch result.



```
shivangi.saklani@passpoli:~/ie709$ python ques25_new.py
30
Arrivals= [0, 0, 12, 24, 36, 48, 60, 72, 84, 96, 108, 120, 132, 144, 156, 168, 180, 192, 204, 216, 228, 240, 252, 264, 276, 288, 300, 312, 324, 336]
we have the following from bailey welch rule:
the average total waiting time of all 30 patients is  388.814545549 minutes
the average idle time of a practitioner in a day is  3.4627732545  minutes
30
Arrivals= [0, 5.95, 13.9, 23.85, 35.8, 47.8, 59.8, 71.8, 83.8, 95.8, 107.8, 119.8, 131.8, 143.8, 155.8, 167.8, 179.8, 191.8, 203.8, 215.8, 227.8, 239.8, 251.8, 263.8, 2
75.8, 287.8, 299.8, 311.8, 323.8, 335.8]
modified rule gives us:
the average total waiting time of all 30 patients is  384.65355077 minutes
the average idle time of a practitioner in a day is  3.41917597381  minutes
shivangi.saklani@passpoli:~/ie709$
```

Figure 8: Implementation of the improvised method which dominates Bailey and Welch's method



```
shivangi.saklani@kanjur:~/ie709$ python ques25_new.py
30
Arrivals= [0, 0, 12, 24, 36, 48, 60, 72, 84, 96, 108, 120, 132, 144, 156, 168, 180, 192, 204, 216, 228, 240, 252, 264, 276, 288, 300, 312, 324, 336]
we have the following from bailey welch rule:
the average total waiting time of all 30 patients is  388.814545549 minutes
the average idle time of a practitioner in a day is  3.4627732545  minutes
30
Arrivals= [0, 6, 14, 24, 36, 48, 60, 72, 84, 96, 108, 120, 132, 144, 156, 168, 180, 192, 204, 216, 228, 240, 252, 264, 276, 288, 300, 312, 324, 336]
modified rule gives us:
the average total waiting time of all 30 patients is  380.398388687 minutes
the average idle time of a practitioner in a day is  3.49895917765  minutes
shivangi.saklani@kanjur:~/ie709$
```

Figure 9: Implementation of the improvised method with starting IAT=6, turns out to be close to Bailey and Welch's method