



# **Ubiquitous Sensing and Smart City**

## **Assignment Three**

**By:**

**Aisha Hagar**

**Khadija Hesham**

**Mohammed Elnamory**

**Supervised by Prof. Burak Kantarci**

## 1. Setup.txt

In the configuration file 'Setup.txt' we changed the '|Days of simulation|' and '|Number of users|' according to the assignment requirements.

```
*****SETUP SIMULATION*****
|Days of simulation| = 2

*****USERS*****
|Number of users| = 8000
```

## 2. ZFMTasks.py

- a. In the file 'ZFMTasks.py' we modified 'task\_generator()' function to generate the tasks according to the requirements in the 'Table1':

Table 1 Task generation requirements

Task Feature	Requirement
Day	Distribution consistent in [1, 2]
Hour	40%: 9:00 AM-11:00 AM 30%: 12:00 PM-5:00 PM 30%: 6:00 PM-8:00 AM
Duration (minutes)	50% in {20, 40, 60} 40% in {30, 50, 70} 10% in {10, 80, 100}
Task Value	Uniformly distributed in [1,10]

```
for i in range(num_tasks):
    #day = np.random.choice(range(1, days + 1))
    day = np.random.randint(1, days+1)

    #Task hour and minute
    distrib = np.random.randint(1, 100)
    if distrib <= 40:
        h = np.random.randint(9, 10)
    elif distrib <= 70:
        h = np.random.randint(12, 16)
    else:
        h = np.random.randint(18, 19)
    m = np.random.randint(0, 59)

    # Task duration
    distrib = np.random.randint(1, 100)
    if distrib <= 50:
        dur = np.random.choice([20, 40, 60])
    elif 5 < distrib <= 90:
        dur = np.random.choice([30, 50, 70])
    else:
        dur = np.random.choice([10, 80, 100])

    task_value = round(np.random.uniform(1, 10))
```

- b. We added 'task value' feature to be written in 'mytask.txt'.

[illegible]

- c. The tasks are then generated by running the python file.

The image shows two screenshots of a Linux terminal window. The top screenshot shows the initial startup of the 'CrowdSenSim Simulator'. It displays progress bars for creating a list of events (100%) and downloading a map (100%). It prompts the user to choose from a menu: 1. Use a saved list of events, 2. Create new list of events, 3. Manage saved list, 4. Change User or Days, 0. Quit. The user has entered '> 2'. The bottom screenshot shows the continuation of the simulation setup. It asks for the city name ('luxembourg city'), antenna choice (1), number of days (2), and number of users (8000). It then shows progress bars for elaborating the map (100%) and calculating average edge length (100%). Finally, it displays the number of nodes after the algorithm (236670) and the total algorithm time (18.84935188293457 seconds).

/ID-Task/	/-Lat/	/-Long/	/-Day/	/-Hour/	/-Minute/	/-Duration/	/-Remaining time/	/-Resources/	/-Coverage/	/-Legitimacy/	/-on peak hour/	/-grid_number/	/-Task value			
1	49.607562245147925			6.120559519020145		2	9	56	60	60	6	100	True	True	96	2
2	49.59373052664449			6.119308016672075		1	9	11	40	60	4	100	True	True	51	8
3	49.62053606305832			6.151942448180133		1	9	2	50	50	2	100	True	True	130	9
4	49.61235558816937			6.133935046920279		1	9	56	20	20	3	100	True	True	98	6
5	49.62861524673394			6.162366753876891		2	12	2	50	50	4	100	True	False	146	9
6	49.63199562482738			6.173192302632164		1	9	15	60	60	6	100	True	True	162	5

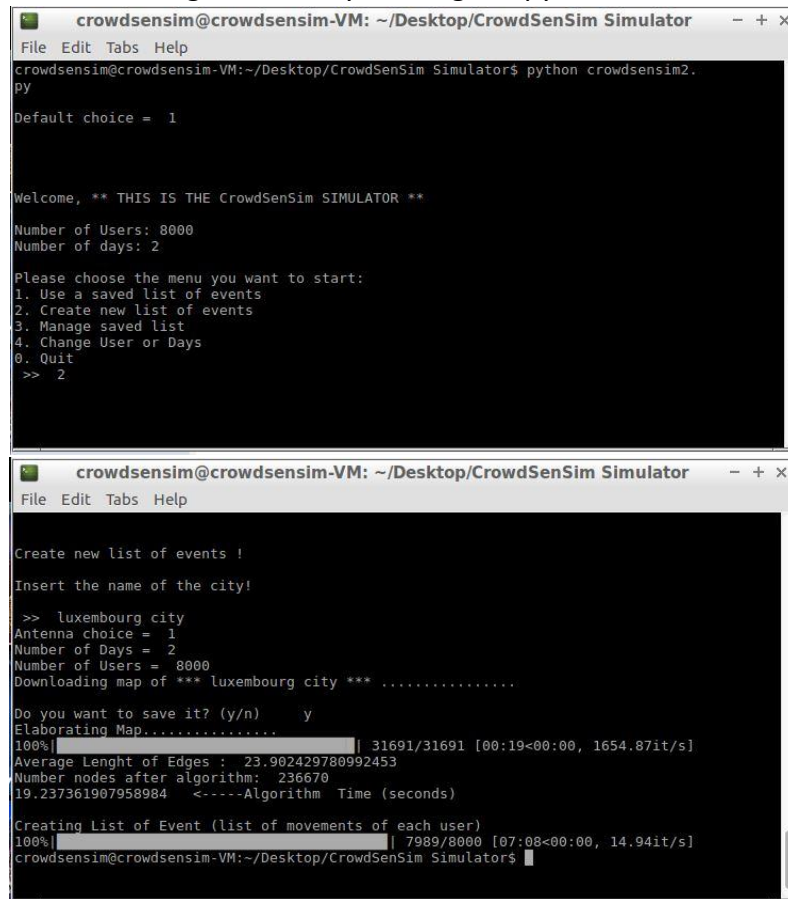
### 3. Crowdsensim2.py

- a. In 'crowdsensim2.py' we modified the code section responsible for deciding the route, to decide the route stochastically.

```
(length, path)= nx.single_source_dijkstra(G_old, origin_node, target=None, cutoff=cutoff, weight='length')
ids=[]
idr=-1
for l in length:
    if length[l]>cut:
        ids.append(l)
try:
    p=np.random.dirichlet(ids)
    idr=np.random.choice(ids, p=p)
except (IndexError, ValueError):
    idr=-1
if idr==-1:
    idr=max(length, key=length.get)

#print (cut-length[idr])
##route = nx.shortest_path(G_old, origin_node, destination_node)
route=path[idr]
```

- b. The movements are then generated by running the python file.



```
crowdsensim@crowdsensim-VM: ~/Desktop/CrowdSenSim Simulator
File Edit Tabs Help
crowdsensim@crowdsensim-VM:~/Desktop/CrowdSenSim Simulator$ python crowdsensim2.py
Default choice = 1

Welcome, ** THIS IS THE CrowdSenSim SIMULATOR **

Number of Users: 8000
Number of days: 2

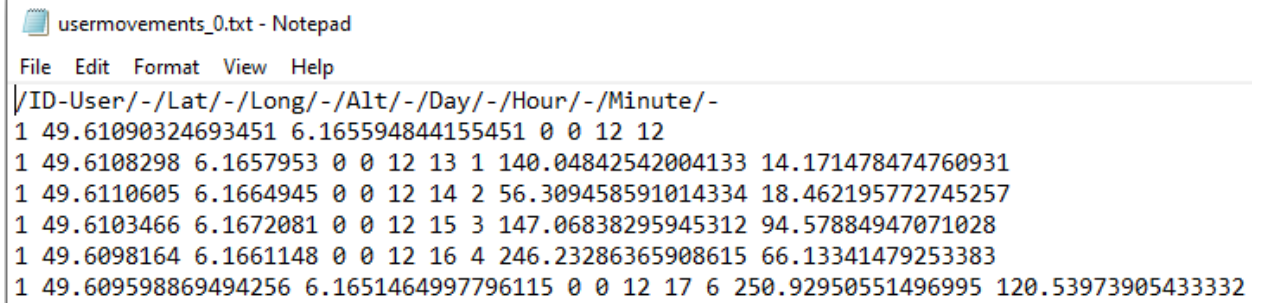
Please choose the menu you want to start:
1. Use a saved list of events
2. Create new list of events
3. Manage saved list
4. Change User or Days
0. Quit
>> 2

Create new list of events !
Insert the name of the city!

>> Luxembourg city
Antenna choice = 1
Number of Days = 2
Number of Users = 8000
Downloading map of *** Luxembourg city *** .....
Do you want to save it? (y/n) y
Elaborating Map..... 31691/31691 [00:19<00:00, 1654.87it/s]
Average Length of Edges : 23.902429780992453
Number nodes after algorithm: 236670
19.237361907958984 <-----Algorithm Time (seconds)

Creating List of Event (list of movements of each user)
100%|.....| 7989/8000 [07:08<00:00, 14.94it/s]
crowdsensim@crowdsensim-VM:~/Desktop/CrowdSenSim Simulator$
```

- c. A list of user movements for 8000 users is generated in 'user\_movements\_0.txt' and 'usermovements\_1.txt'. Below is a snippet from 'usermovements\_0.txt':



```
usermovements_0.txt - Notepad
File Edit Format View Help
/ID-User/-/Lat/-/Long/-/Alt/-/Day/-/Hour/-/Minute/-
1 49.61090324693451 6.165594844155451 0 0 12 12
1 49.6108298 6.1657953 0 0 12 13 1 140.04842542004133 14.171478474760931
1 49.6110605 6.1664945 0 0 12 14 2 56.309458591014334 18.462195772745257
1 49.6103466 6.1672081 0 0 12 15 3 147.06838295945312 94.57884947071028
1 49.6098164 6.1661148 0 0 12 16 4 246.23286365908615 66.13341479253383
1 49.609598869494256 6.1651464997796115 0 0 12 17 6 250.92950551496995 120.53973905433332
```