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))
```

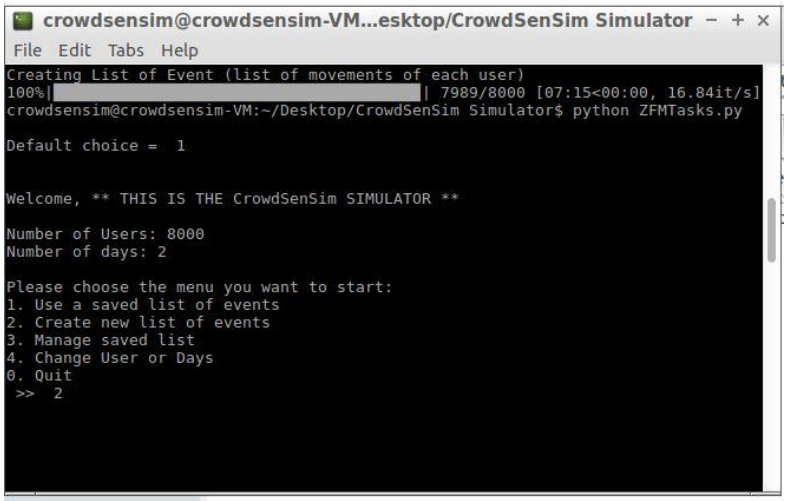
b. We added 'task_value' feature to be written in 'mytask.txt'.

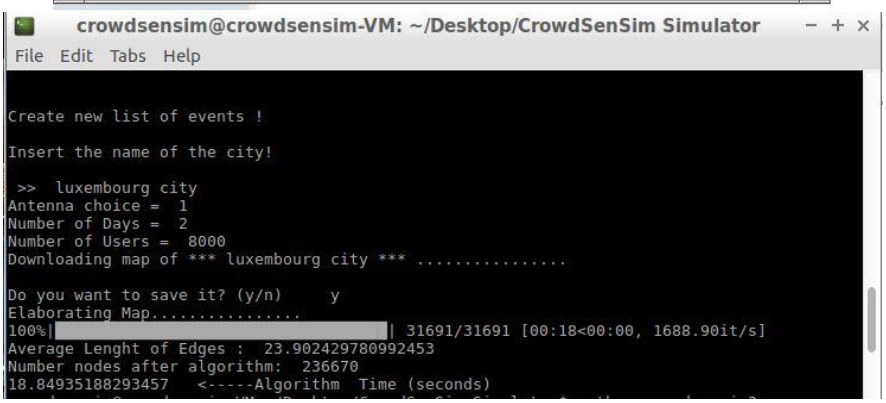
```
        # task txt file
        task_file = open('./Inputs/Mobility/differentradius/50/'+str(run_num)+'/mytask.txt', 'w')
        task_file.write("/ID-Task/  -/Lat/  -/Long/ -/Day/  -/Hour/ -/Minute/   -/Duration/ -/Remaining time/ -/Resources/    -/Coverage/ -/Ligitimacy/ -/on peak hour/
-/grid_number/ -/Task value\n")
        for i in range(0,len(tasks)):
            task_file.write("{}\t {}\t {}\t {}\t {}\t {}\t {}\t {}\t {}\t {}\t {}\t {}\t {}\n".format(tasks[i][0], tasks[i][1], tasks[i][2], tasks[i][3], tasks[i][4],
tasks[i][5], tasks[i][6], tasks[i][7], tasks[i][8], tasks[i][9], tasks[i][10], tasks[i][11], tasks[i][12], tasks[i][13]))

        task_file.close()

        # task csv file
```

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





d. The 1000 generated tasks are saved in 'mytask.txt'. The figure below shows a snippet from 'mytask.txt':

/ID-Task/	-/Lat/	-/Long/	-/Day/	-/Hour/	-/Minute/	-/Duration/	-/Remaining time/	-/Resources/	-/Coverage/	-/Ligitimacy/	-/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	40	8	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

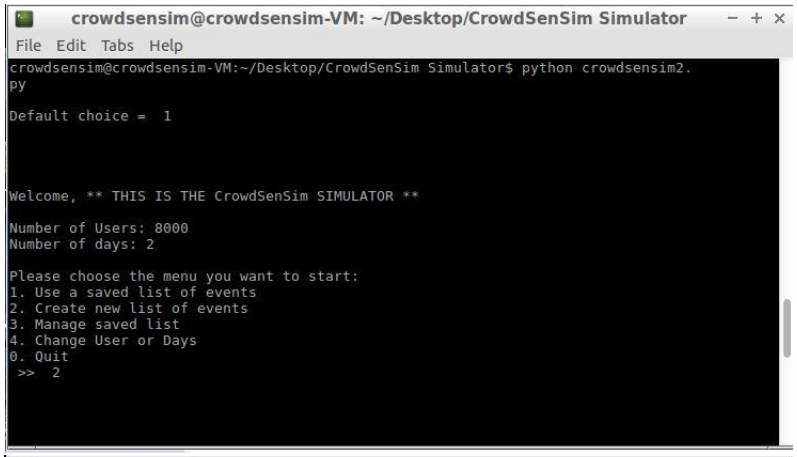
3. Crowdsensim2.py

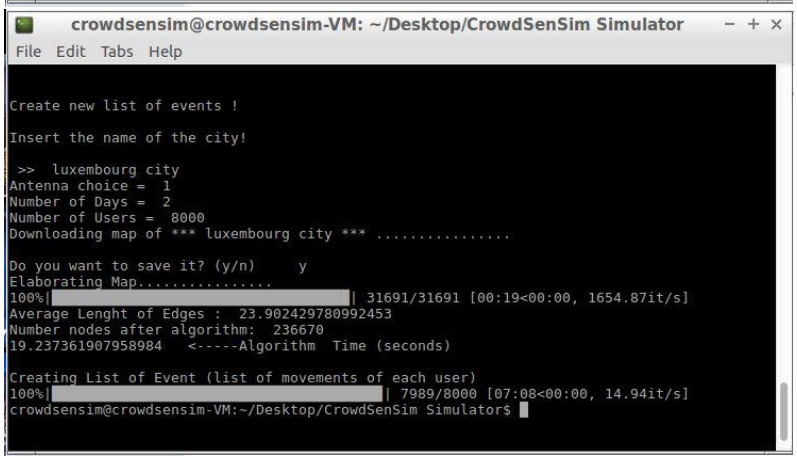
a. In ‘crowdsensim2.py’ we modified the code section responsible for deciding the route, to decide the route stochastically.

```
(length, path)= nx.single_source_dijkstra(G_old, origin_node, target=None, cutoff=cutadded, 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.





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
```