

r4.py

```
1  from utilities_new import ServiceRequest, ServiceType, assign_customers_to_best_van,
   Van
2  import random
3  import networkx as nx
4  import matplotlib.pyplot as plt
5
6  seed=1000          # seed the graph for reproducibility, you should be doing this
7  G= nx.gnp_random_graph (100, .3, seed=seed )          # here we create a random binomial
   graph with 10 nodes and an average (expected) connectivity of 10*.3= 3.
8  nx.is_connected(G)
9
10 for u, v in G.edges:      # needed for requirement R3.
11     G.add_edge(u, v, weight=round(random.random(),1))
12
13 #make 30 vans, all start at node 0
14 vans = []
15 for i in range(1,61):
16     van = Van(i)
17     van.route.append(0)
18     vans.append(van)
19
20 customer_id = 0
21 clocktick = 0
22 while clocktick < 600: #check for time, 600 clock ticks = 600min = 10hrs = runtime
   for simulation
23     print("CLOCKTICK " + str(clocktick))
24     clocktick += 1
25
26     unassigned_service_requests = []
27     #randomize 10 requests per clock tick, this makes 600 requests per hour
28     #unassigned_service_requests.append(ServiceRequest(1, ServiceType.Pickup, 8))
29     for i in range(0, 10):
30         unassigned_service_requests.append(ServiceRequest(customer_id,
   ServiceType.Pickup, random.randint(0,99)))
31         unassigned_service_requests.append(ServiceRequest(customer_id,
   ServiceType.Dropoff, random.randint(0,99)))
32         customer_id += 1
33
34     # Perform any pickups or dropoffs
35     for van in vans:
36         van.pickup_or_dropoff()
37
38     assign_customers_to_best_van(vans, unassigned_service_requests, G)
39
40     # Sort van service queues
41     for van in vans:
42         van.sort_service_queue2(G)
```

```
43
44     # Move vans to next nodes
45     for van in vans:
46         van.move_to_next_node(G)
47
48
49 while True:
50     print("end run")
51     # Perform any pickups or dropoffs
52     for van in vans:
53         van.pickup_or_dropoff()
54
55     assign_customers_to_best_van(vans, unassigned_service_requests, G)
56
57     # Sort van service queues
58     for van in vans:
59         van.sort_service_queue2(G)
60
61     # Move vans to next nodes
62     for van in vans:
63         van.move_to_next_node(G)
64
65     empty_count = 0
66     for van in vans:
67         if len(van.queue) == 0:
68             empty_count += 1
69
70     if empty_count == len(vans):
71         break
72
73     total_distance = 0
74     total_trips = 0
75     for van in vans:
76         total_distance += van.distance_travelled
77         total_trips += van.trips_taken
78
79     average_distance = total_distance / len(vans)
80
81     print(f"Average Distance Travelled: {average_distance}")
82     print(f"Total Trips Taken: {total_trips}")
83
```