Plot4GMNS

A visualization tool for visualizing and analyzing transportation network and demand files in GMNS format.

Plot4GMNS is an open-source transportation network data visualization and analysis tool based on GMNS data format. By taking advantage of OSM2GMNS and Grid2demand tools to obtain routable transportation network and demand, Plot4GMNS aims to visualize and analyze the node, link, POI, agent, and zone data of the network.

1. Data Format

node.csv

name	
node_id	*
osm_node_id	
osm_highway	
zone_id	
ctrl_type	*
node_type	
activity_type	*
is_boundary	
x_coord	*
y_coord	*
main_node_id	
poi_id	
notes	
activity_zone_id	
production	*
attraction	*
activity_location_tab	

link.csv

name	
link_id	
osm_way_id	
from_node_id	*
to_node_id	*
dir_flag	
length	*
lanes	
free_speed	*
capacity	*
link_type_name	*
link_type	*
geometry	*
allowed_uses	*
from_biway	

poi.csv

name	
poi_id	
osm_way_id	
osm_relation_id	
building	*
amenity	
way	
geometry	*
centroid	*
area	
area_ft2	
activity_zone_id	*

demand.csv

o_zone_id	*
o_zone_name	*
d_zone_id	*
d_zone_name	*
accessibility	
volume	*
geometry	*

input_agent.csv

agent_id	
agent_type	
o_node_id	*
d_node_id	*
o_osm_node_id	
d_osm_node_id	
o_zone_id	*
d_zone_id	*
geometry	*
departure_time	*

poi_trip_rate.csv

poi_type_id	*
building	*
unit_of_measure	
trip_purpose	*
production_rate1	*
attraction_rate1	*
production_notes	

attraction_notes	

zone.csv

activity_zone_id	*
name	*
centroid_x	*
centroid_y	*
geometry	*
centroid	*
total_poi_count	*
residential_poi_count	*
office_poi_count	*
shopping_poi_count	*
school_poi_count	*
parking_poi_count	*
boundary_node_count	*
total_production	*
total_attraction	*

2. How to use (take DC_Downtown dataset for example)

2.1 install

pip install plot4gmns

2.2 build network from csv files

(1) import the package

```
import plot4gmns as pg
```

(2) specify the path to files and read csv files

```
net=pg.readNetwork('./data')
```

2.3 check valid attributes

(1) valid node attributes list

```
net.get_valid_node_attr_list()
```

output:

attr	type
ctrl_type	int
activity_type	str
production	float
attraction	float

(2) valid link attributes list

```
net.get_valid_link_attr_list()
```

output:

attr	type
length	float
lanes	int
free_speed	float
capacity	float
link_type_name	str
allowed_uses	str

(3) valid poi attributes list

```
net.get_valid_poi_attr_list()
```

output:

attr	type
building	str
activity_zone_id	int

(4) valid zone id list

```
net.get_valid_zone_id_list()
```

output:

min zone id	max zone id
1	45

(5) get node attribute value list

```
pg.get_node_attr_value_list(net,'ctrl_type')
```

output:

ctrl_type	number
0	1654
1	208

```
pg.get_node_attr_value_list(net,'production')
```

output:

attr	min value	max value
production	0.0	1000.0

```
pg.get_node_attr_value_list(net,'activity_type')
```

output:

activity_type	number
primary	240
motorway	19
tertiary	104
secondary	160
residential	21

```
pg.get_node_attr_value_list(net,'attraction')
```

output:

attr	min value	max value
attraction	0.0	1000.0

(6) get link attribute value list

```
pg.get_link_attr_value_list(net,'link_type_name')
```

output:

link_type_name	number
primary	482
tertiary	295
motorway	22
secondary	381
residential	43
connector	2564

```
pg.get_link_attr_value_list(net,'allowed_uses')
```

output:

allowed_uses	number
auto	1223
all	2564

```
pg.get_link_attr_value_list(net,'length')
```

output:

attr	min value	max value
length	0.7308094767160183	318.586173324247

```
pg.get_link_attr_value_list(net,'lanes')
```

output:

attr	min value	max value
lanes	1	5

```
pg.get_link_attr_value_list(net,'capacity')# if the capacity value exists in the link.csv
```

output:

```
the 'capacity' attribute value missing
```

(6) get POI attribute value list

```
pg.get_poi_attr_value_list(net,'building')
```

output:

building	number
yes	999
courthouse	1
place_of_worship	9
house	5
commercial	13
arts_centre	1
office	35
university	12

```
pg.get_poi_attr_value_list(net,'activity_zone_id')
```

output:

activity_zone_id	number
25	18
20	61
10	118
15	49
14	80
19	27
5	56
9	43
13	58

(7) get zone id value list

```
pg.get_zone_id_list(net,zone_id=1)
```

output:

zone_id	name	total_poi_count	total_production	total_attraction
1	A1	96.0	844.0478253270337	684.8038331279233

```
pg.get_zone_id_list(net)
```

output:

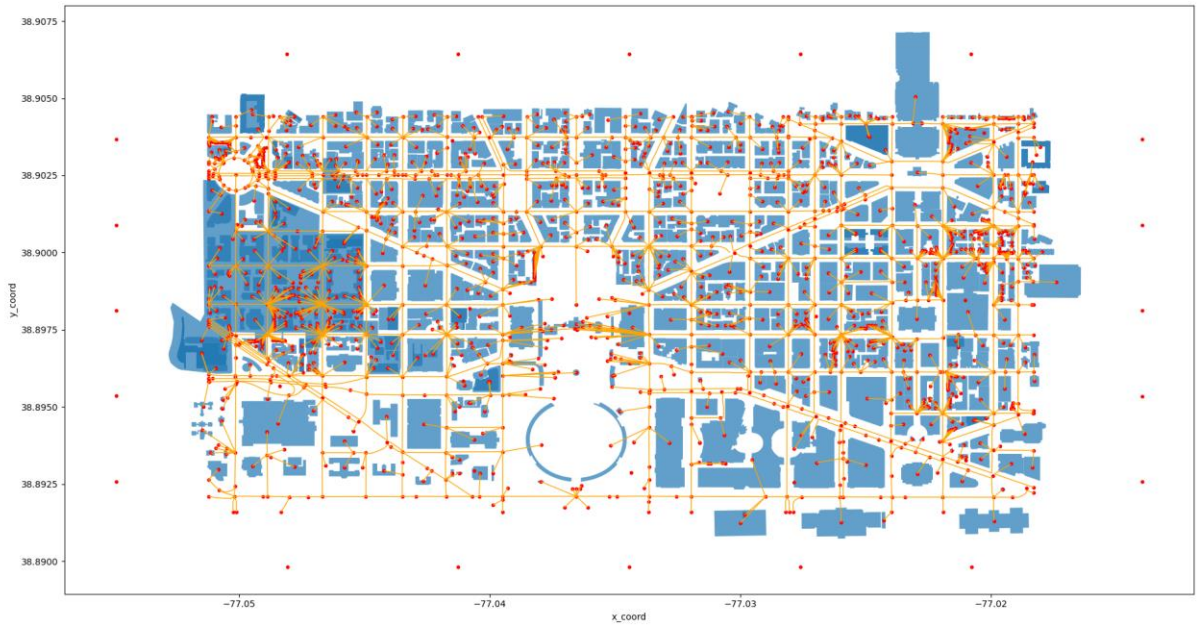
zone_id	name	total_poi_count	total_production	total_attraction
1	A1	96.0	844.0478253270337	684.8038331279233
2	A2	44.0	742.3576762712639	668.437867088064
3	A3	54.0	729.8692152026176	703.8141044522879
4	A4	49.0	596.3070090494338	654.1352135049601
5	A5	56.0	452.5258671580673	388.14464787057284
6	B1	52.0	973.3981069754944	698.8563163042176

2.4 show network of different modes

(1) show network of different modes (all,auto, bike, walk, rail)

```
pg.showNetByAllMode(net)
```

output:




```
pg.showNetByAutoMode(net)
```

output:


```
# if the mode exists in the link.csv
pg.showNetByBikeMode(net)
pg.showNetByWalkMode(net)
pg.showNetByRailMode(net)
```

2.5 show network by node attributes

Only valid node attributes and values can be displayed. So, the first step is to check the valid node attribute fields and values in the network.

- net.get_valid_node_attr_list()
- pg.get_node_attr_value_list(net,attr)

(1) show network by node ‘ctrl_type’

```
pg.showNetByNodeAttr(net,{'ctrl_type':1})
```

output:


```
pg.showNetByNodeAttr(net,{'ctrl_type':(0,1)})
```

output:

(2) show network by node activity_type

```
pg.showNetByNodeAttr(net,{'activity_type':'primary'})
```

output:


```
pg.showNetByNodeAttr(net,{'activity_type':['primary', 'secondary']})
```

output:

(3) show network by node attraction range

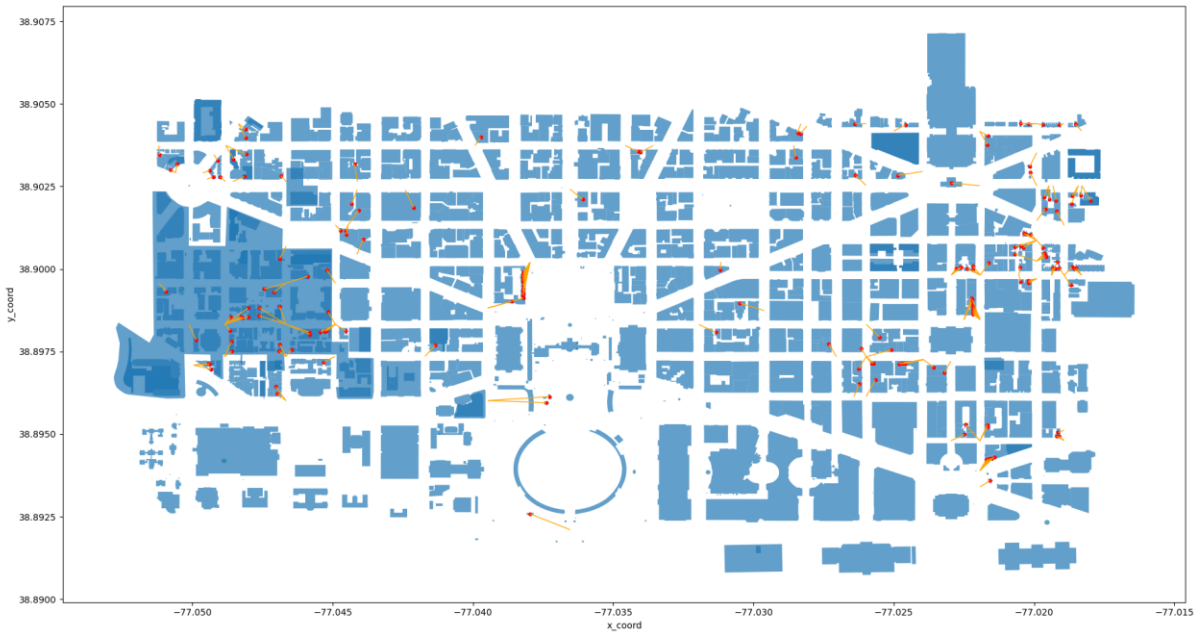
```
pg.showNetByNodeAttr(net,{'attraction':(0,1)})
```

output:

(4) show network by node production range

```
pg.showNetByNodeAttr(net,{'production':(1,2)})
```

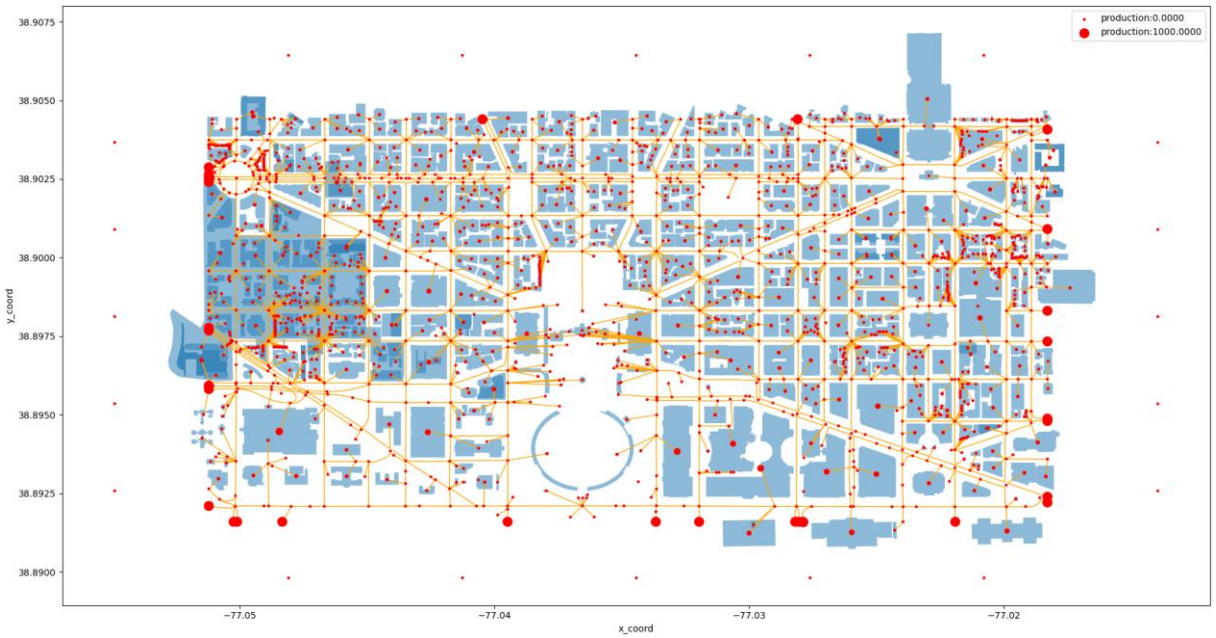
output:



(5) show network by node production value with graduated symbology in size

```
pg.showNetByNodeProduction(net)
```

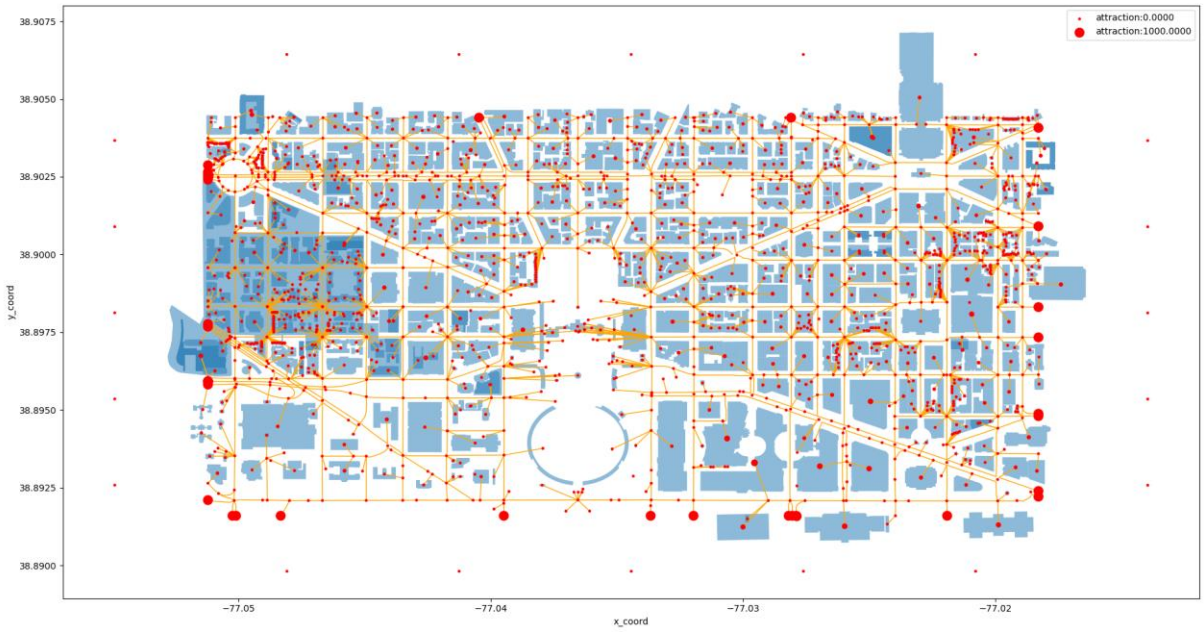
output:



(6) show network by node attraction value with graduated symbology in size

```
pg.showNetByNodeAttraction(net)
```

output:



2.6 show network by link attributes

Only valid link attributes and values can be displayed. So, the first step is to check the valid link attribute fields and values in the network.

- net.get_valid_link_attr_list()

- pg.get_link_attr_value_list(net,attr)

(1) show network by link type name

```
pg.showNetByLinkAttr(net,{'link_type_name':'secondary'})
```

output:


```
pg.showNetByLinkAttr(net,{'link_type_name':['secondary','primary']})
```

output:

(2) show network by allowed vehicle type on the link

```
pg.showNetByLinkAttr(net,{'allowed_uses':'auto'})
```

output:

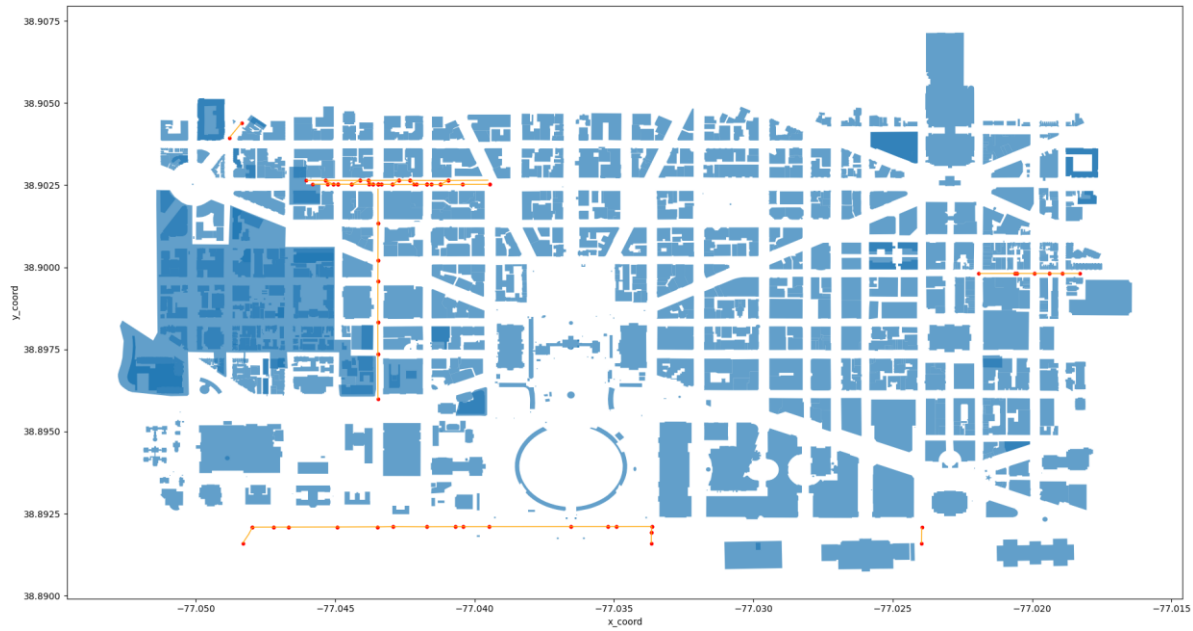

```
pg.showNetByLinkAttr(net,{'allowed_uses':['all','auto']})
```

output:

(3) show network by link free speed range

```
pg.showNetByLinkAttr(net,{'free_speed':(20,40)})
```

output:



(4) show network by link capacity range

```
# if capacity value exists in the link.csv
pg.showNetByLinkAttr(net,{'capacity':(20,40)})
```

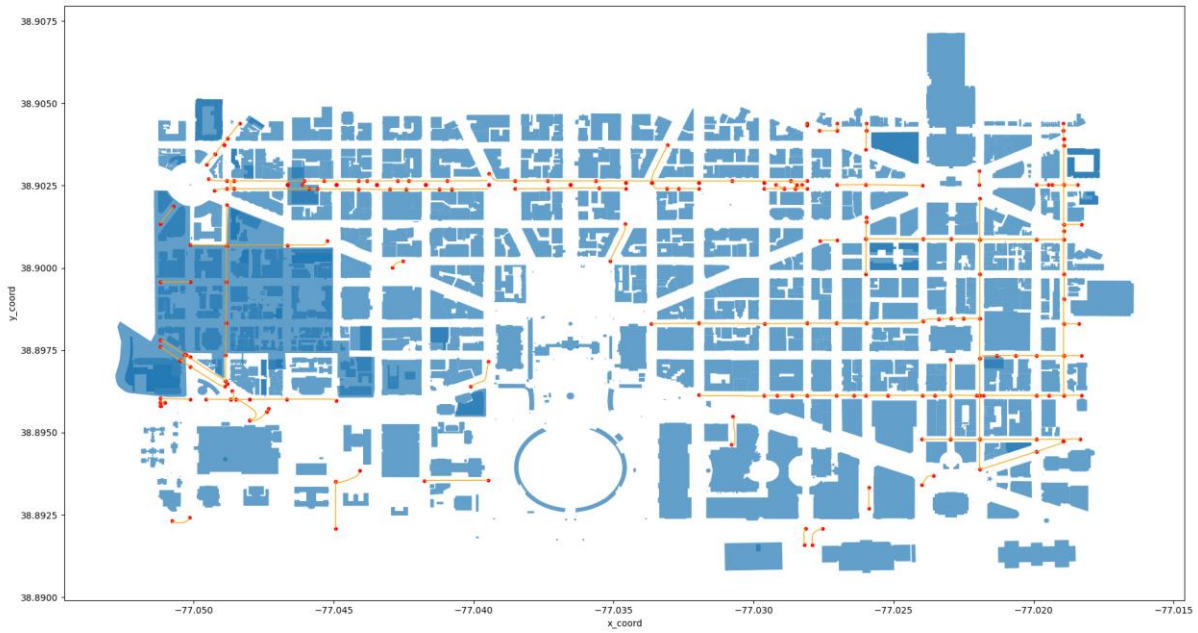
output:

```
There is no link data that meets the requirements
```

(5) show network by number of lanes

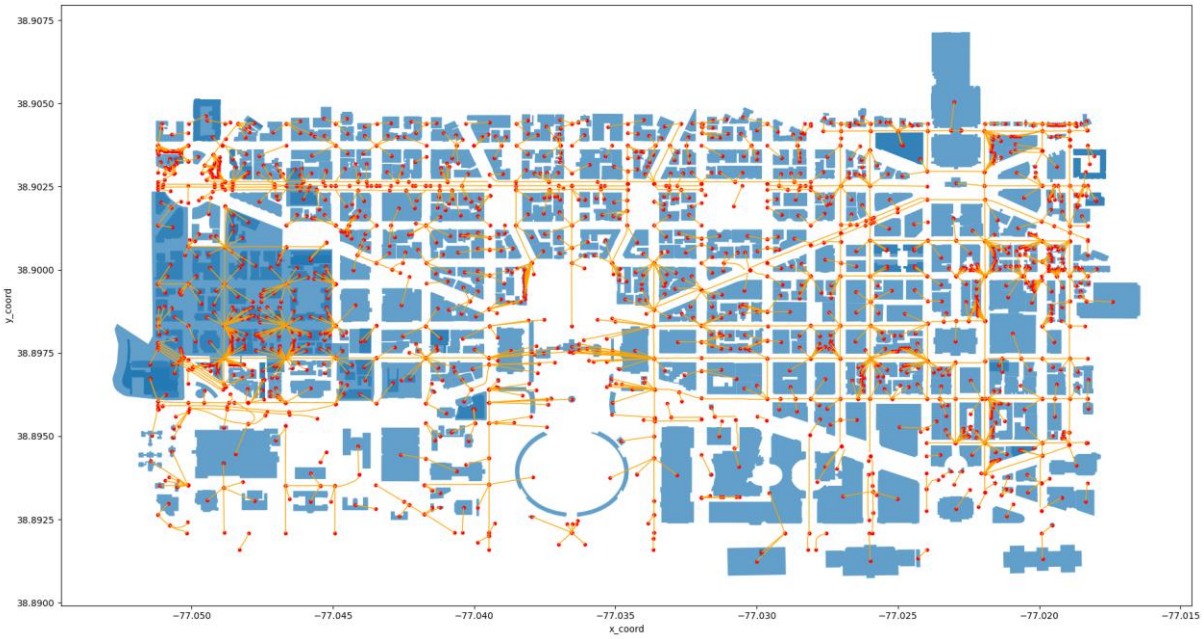
```
pg.showNetByLinkAttr(net,{'lanes':(1,1)})
```

output:



```
pg.showNetByLinkAttr(net,{'lanes':(1,2)})
```

output:



(6) show network by link length range

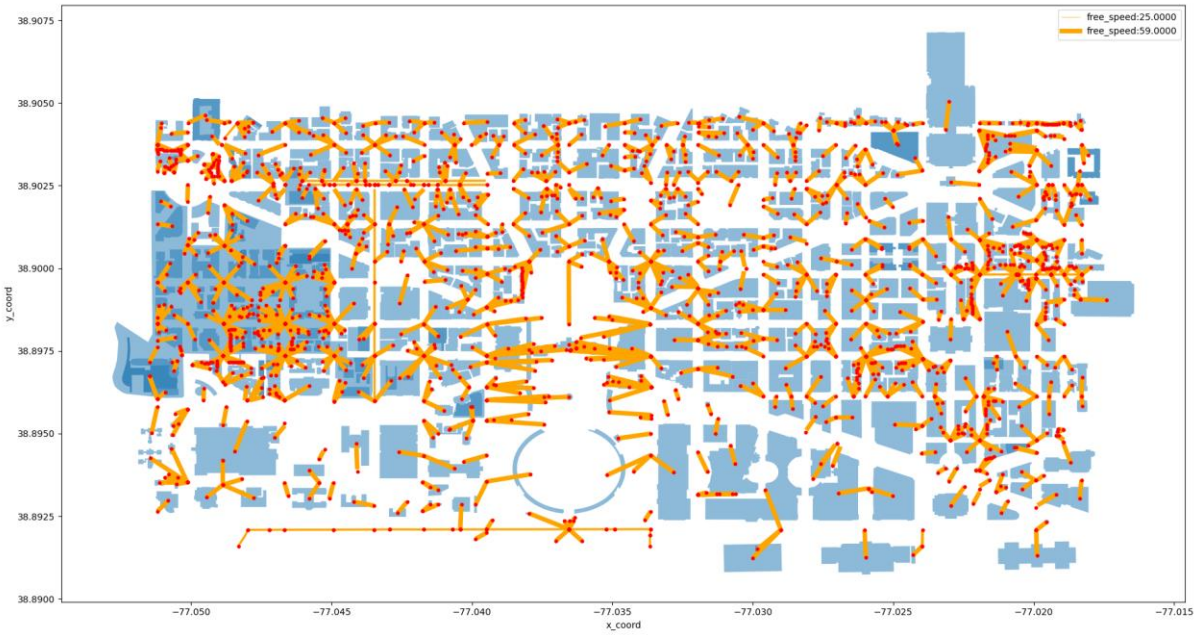
```
pg.showNetByLinkAttr(net,{'length':(100,300)})
```

output:

(7) show network by link free speed with graduated symbology in size

```
pg.showNetByLinkFreeSpeed(net)
```

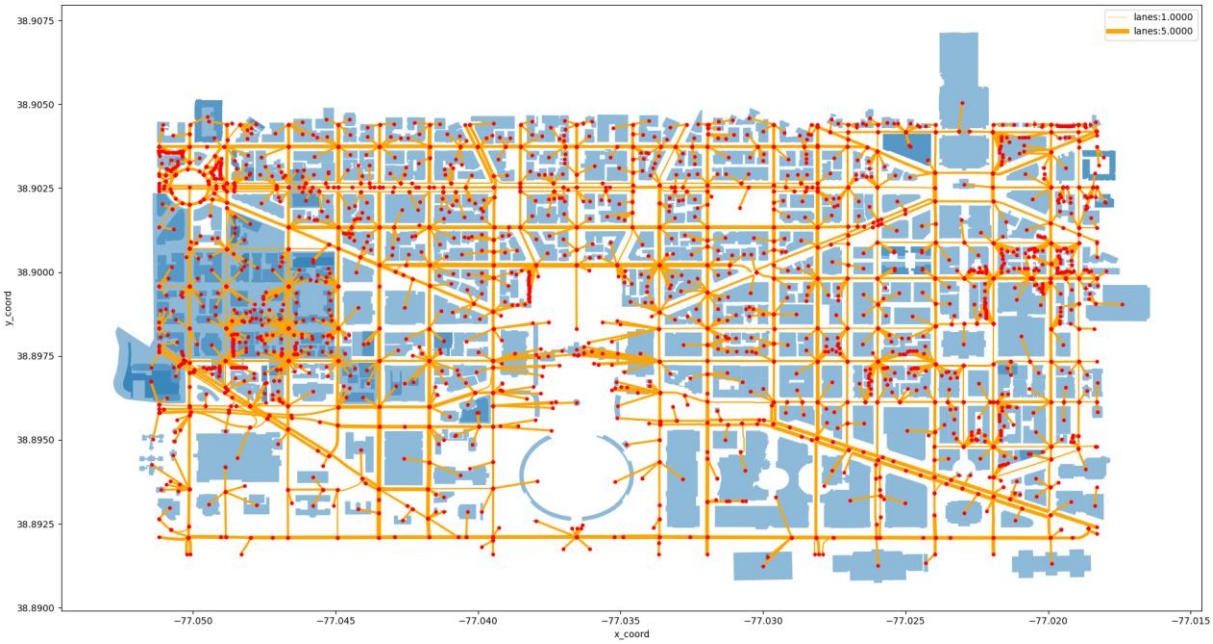
output:



(8) show network by link lanes with graduated symbology in size

```
pg.showNetByLinkLaneNum(net)
```

output:



(9) show network by link capacity with graduated symbology in size

```
# if capacity value exists in the link.csv
pg.showNetByLinkCapacity(net)
```

output:

```
the 'link capacity' attribute value missing
```

2.7 show network by POI attributes

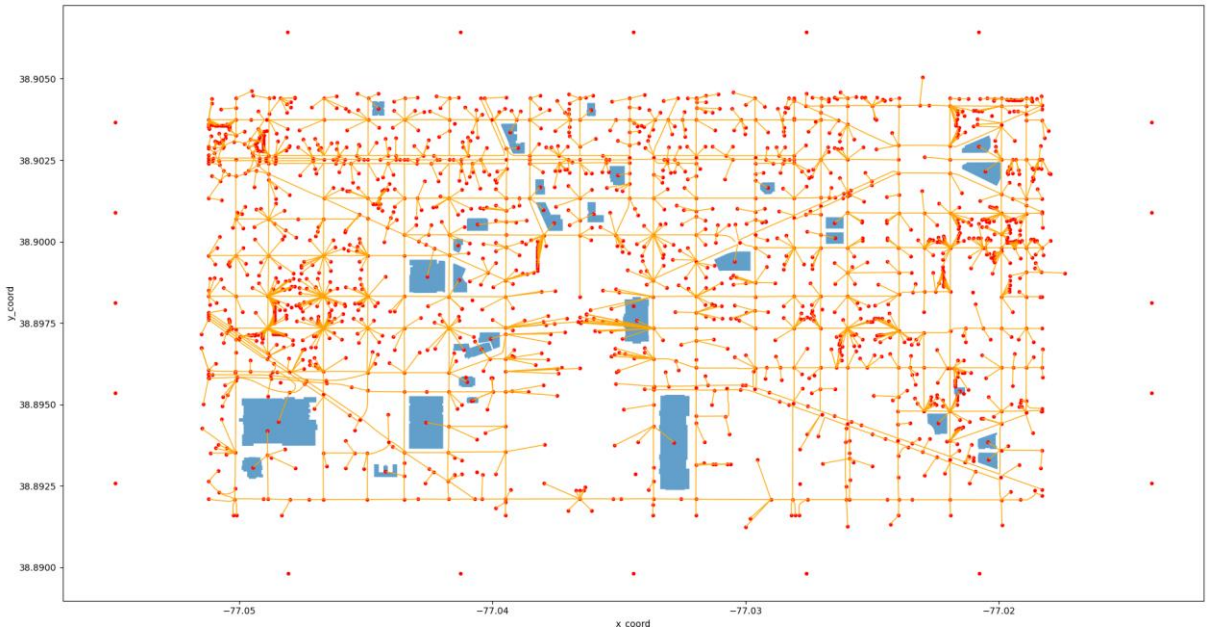
Only valid POI attributes and values can be displayed. So, the first step is to check the valid POI attribute fields and values in the network.

- net.get_valid_poi_attr_list()
- pg.get_poi_attr_value_list(net,attr)

(1) show network by POI building type

```
pg.showNetByPOIAttr(net,{'building':'office'})
```

output:



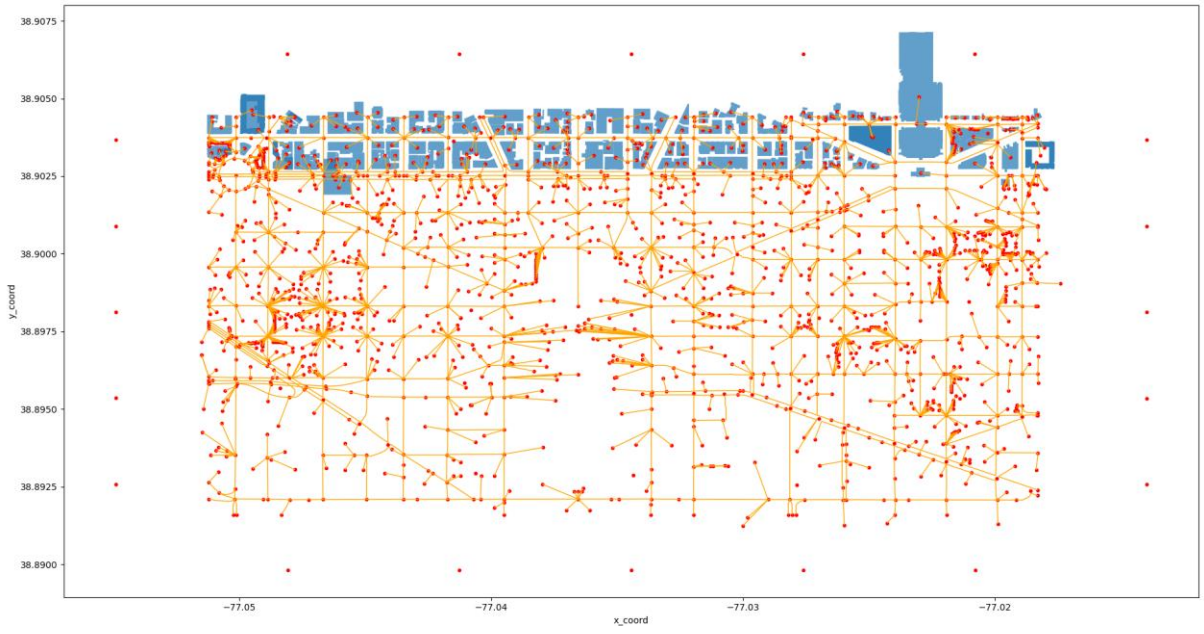

```
pg.showNetByPOIAttr(net,{'building':['office', 'yes']})
```

output:

(2) show network by POI activity_zone_id range

```
pg.showNetByPOIAttr(net,{'activity_zone_id':(1,5)})
```

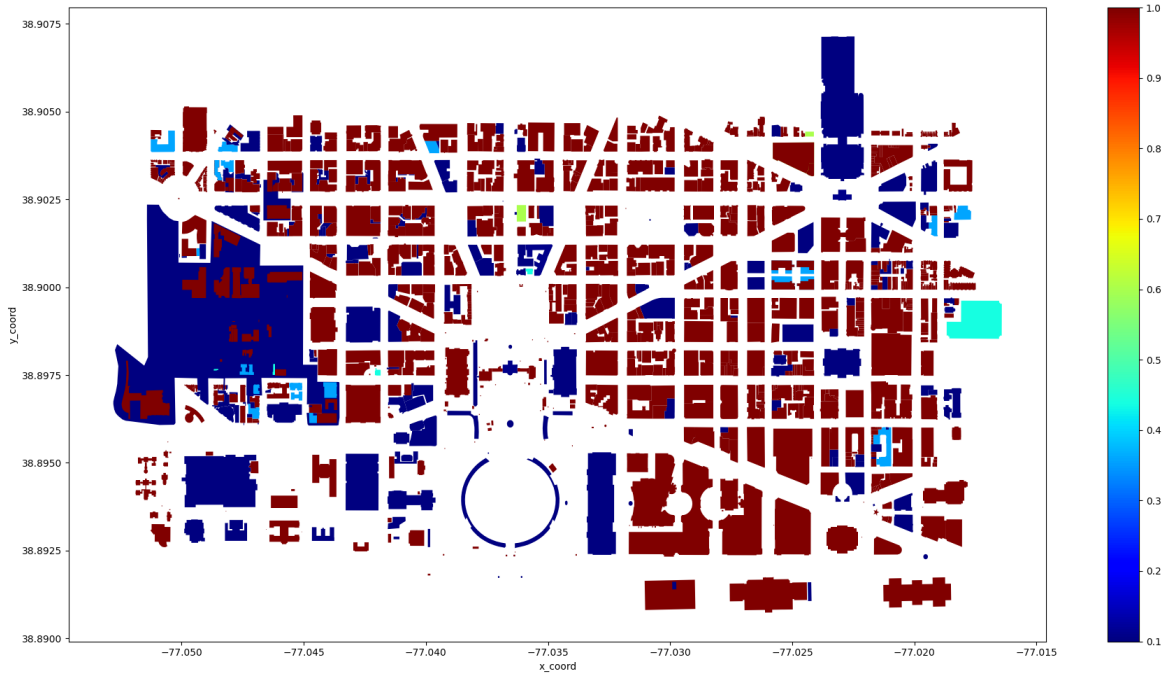
output:



(3) show network by POI attraction heat

```
pg.showNetByPOIAttractionHeat(net)
```

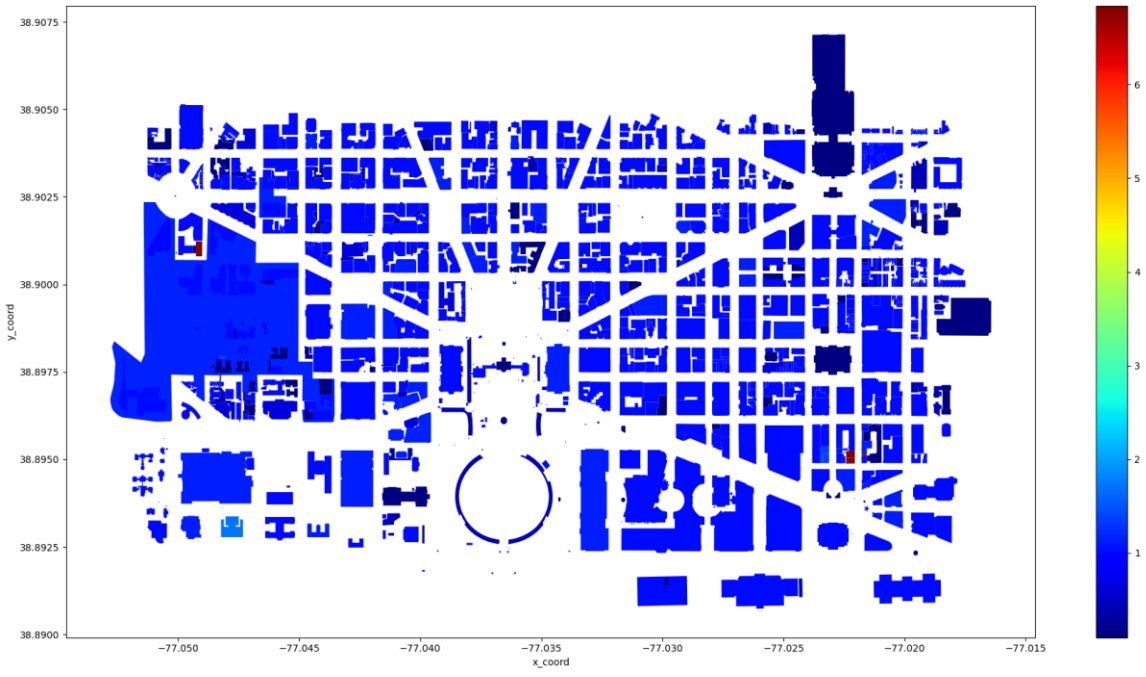
output:



(4) show network by POI production heat

```
pg.showNetByPOIProductionHeat(net)
```

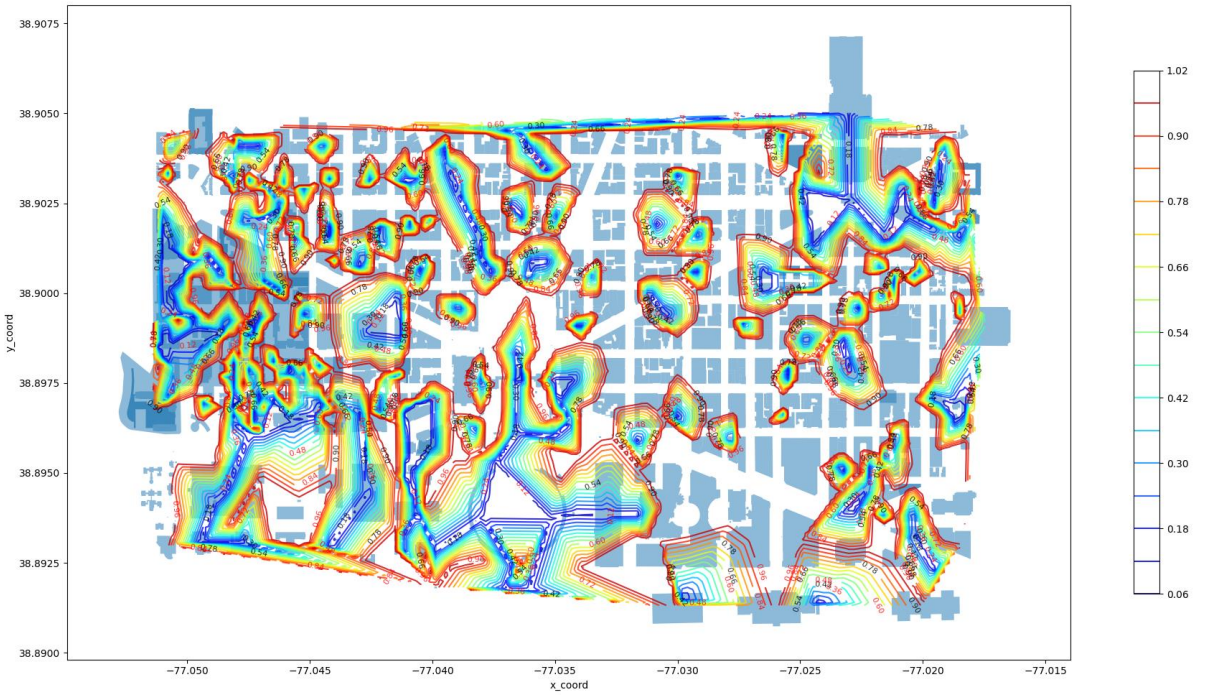
output:



(5) show network by POI attraction contour

```
pg.showNetByPOIAttractionContour(net)
```

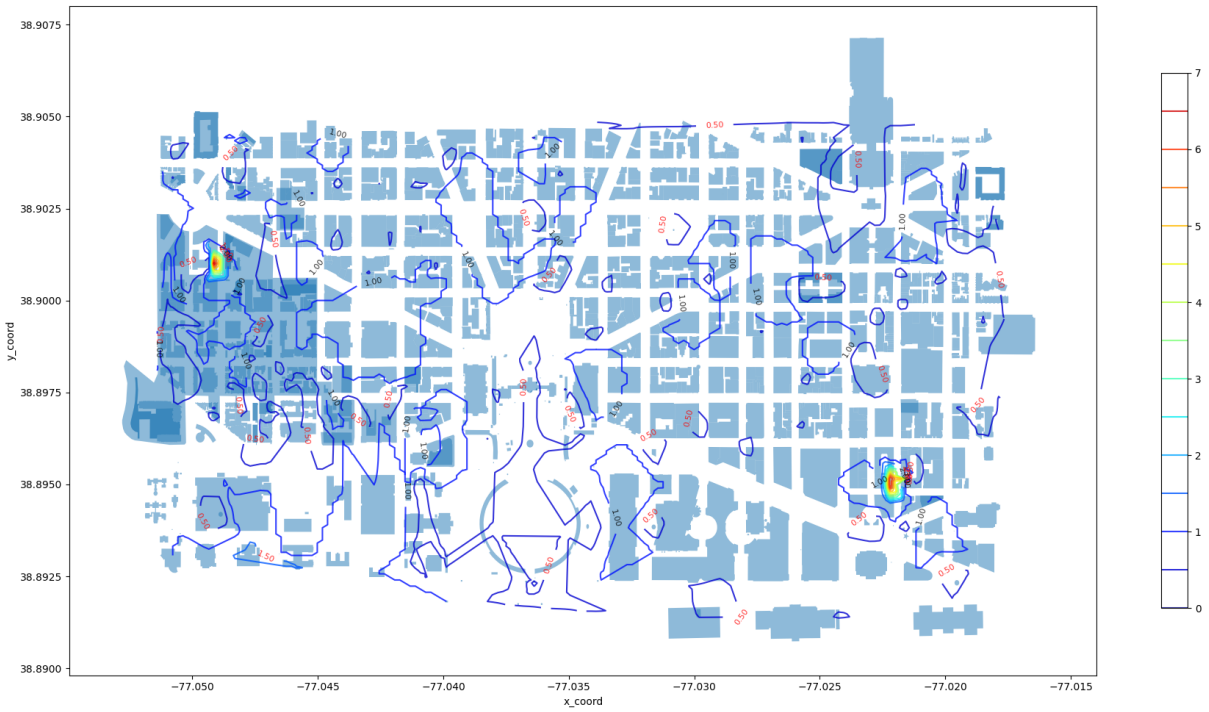
output:



(6) show network by POI production contour

```
pg.showNetByPOIProductionContour(net)
```

output:



2.8 show network by zone attributes

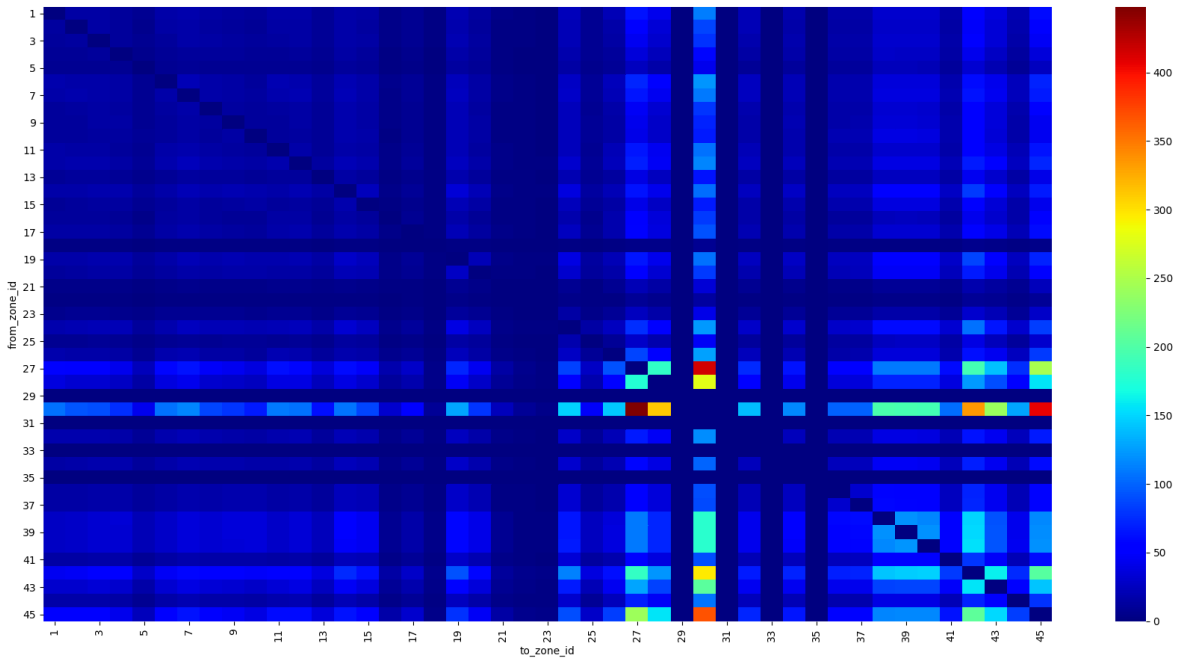
Only valid zone attributes and values can be displayed. So, the first step is to check the valid zone attribute fields and values in the network.

- net.get_valid_zone_id_list()
- pg.get_zone_id_list(net,zone_id)

(1) show network by zone-to-zone demand heat

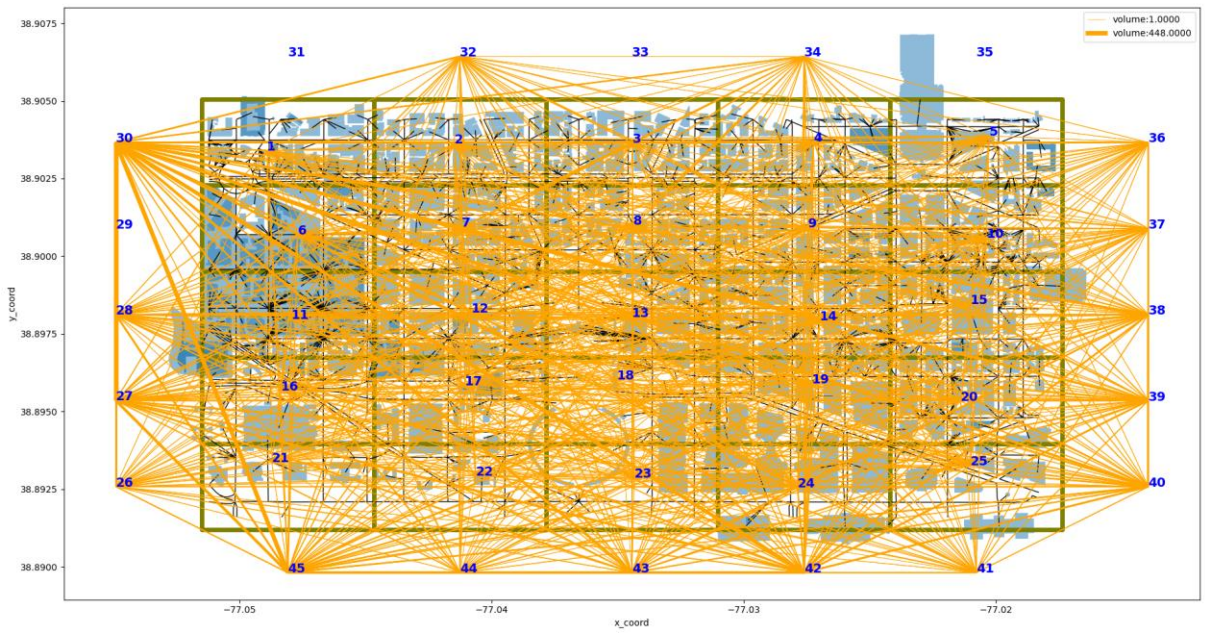
```
pg.showNetByZoneDemandHeat(net,annot=False)
# annot:bool,whether or not show zone-to-zone demand value
```

output:



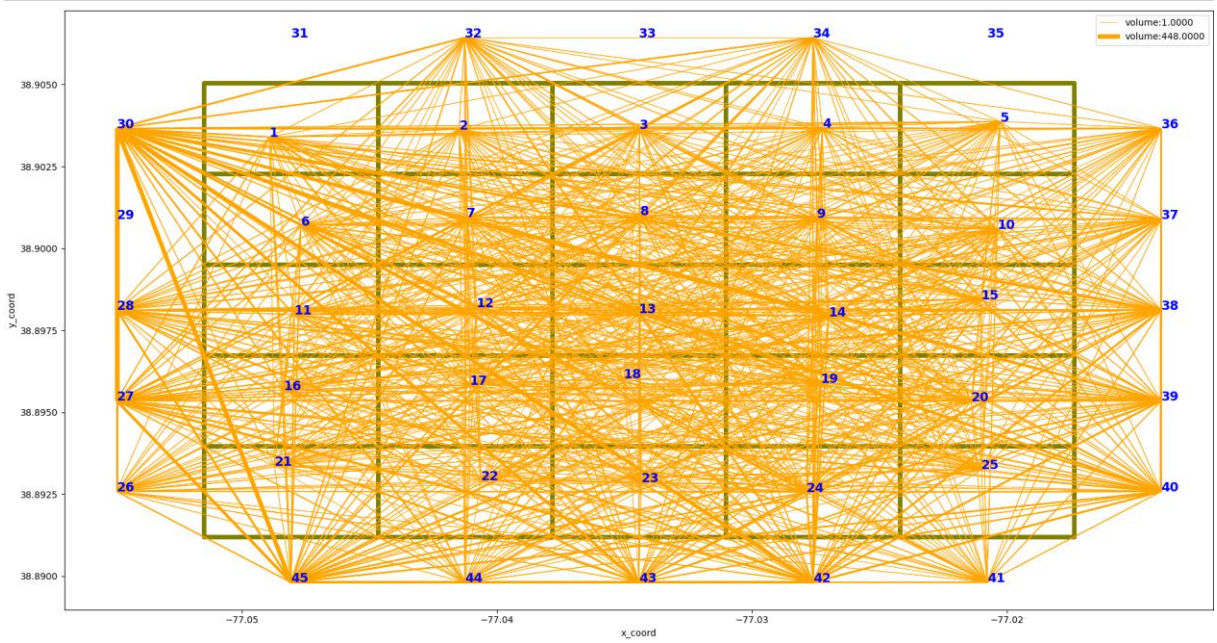
(2) show network by zone demand flow

```
pg.showNetByZoneDemandFlow(net)
```



```
pg.showNetByZoneDemandFlow(net,annot=True,bg=False)
#annot：bool, whether or not show zone id
#bg：bool, whether or not show network (node, link, and poi)
```

output:



(3) show network by node-to-node agents for one or more zone-to-zone O/D pairs

```
pg.showNetByZoneAgent(net,(1,15))
```

output:


```
pg.showNetByZoneAgent(net,[(1,15),(6,5)])
```

output:


(1, 15)agents:14
(6, 5)agents:8
1
5
6
15
38.9075
38.9050
38.9025
38.9000
38.8975
38.8950
38.8925
38.8900
y_coord
−77.050
−77.045
−77.040
−77.035
−77.030
−77.025
−77.020
−77.015
x_coord