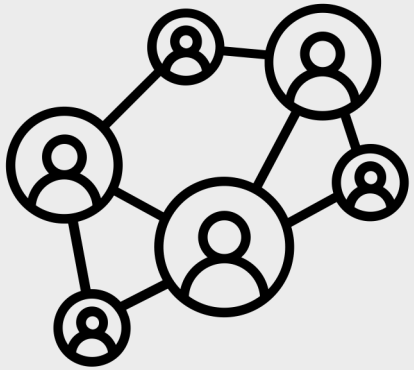


ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY

Sixth Week

Transportation Network

6.1 Local Clustering Coefficient

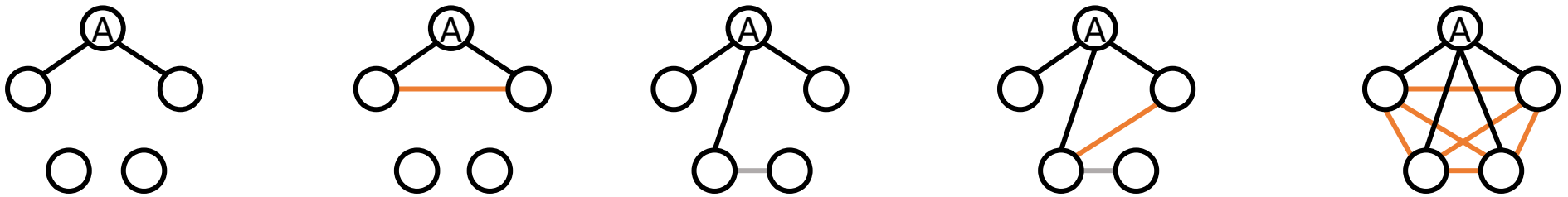
Definition 6.1.1 (Local Clustering Coefficient)

Local clustering coefficient is fractiona of pairs of the node's friends that are friends with each other.

$$C_i = \frac{(\text{number of pairs of neighbors of } i \text{ that are connected})}{(\text{number of pairs of neighbors of } i)}$$

That is, to calculate C_i we go through all distinct pairs of vertices that are neighbors of i in the network, count the number of such pairs that are connected to each other, and divide by the total number of pairs, which is $\frac{1}{2}k_i(k_i - 1)$ where k_i is the degree of i,. C_i is sometimes called the local clustering coefficient and it represents the average probability that a pair of i's friends are friends of one another.

Example



$C_A = \frac{0}{{}_2C_2} = \frac{0}{1}$ $C_A = \frac{1}{{}_2C_2} = \frac{1}{1}$ $C_A = \frac{0}{{}_3C_2} = \frac{0}{3}$ $C_A = \frac{1}{{}_3C_2} = \frac{1}{3}$ $C_A = \frac{6}{{}_4C_2} = \frac{6}{6}$

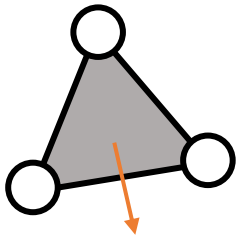
Edges connected to A Edges between neighbors of A Other Edges

6.1 Local Clustering Coefficient

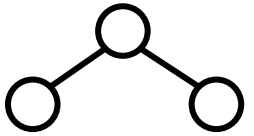
Definition 6.1.2 (Global Clustering Coefficient)

Global clustering coefficient is the mean of the local clustering coefficients for each vertex:

Definition 6.1.3 (Transitivity)



Closed Triangle



Open Triad

$$\frac{3\times(\text{number of triangles})}{(\text{number of open triad})}$$

Definition 6.1.4 (Distance)

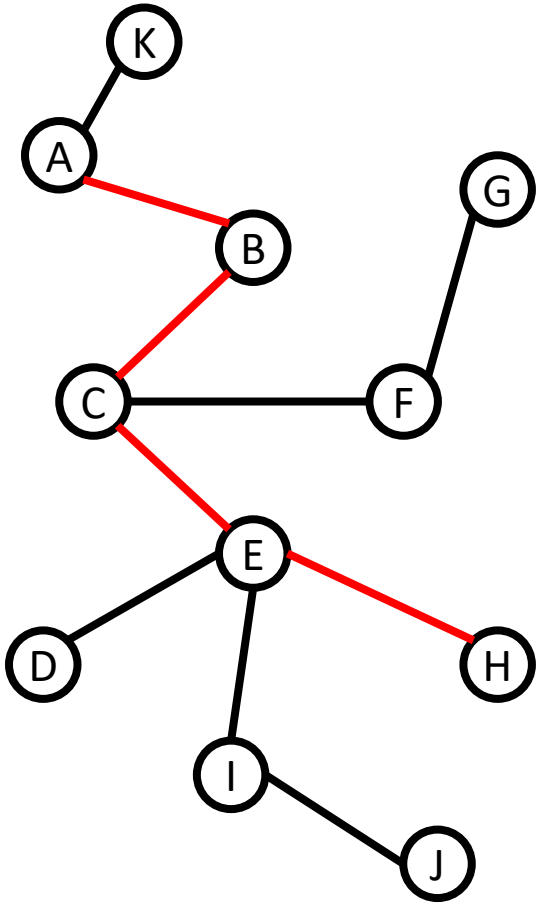
Distance between two nodes is the lenght of shortest path between them.

6.1 Local Clustering Coefficient

Definition 6.1.4 (Distance)

Distance between two nodes is the lenght of shortest path between them.

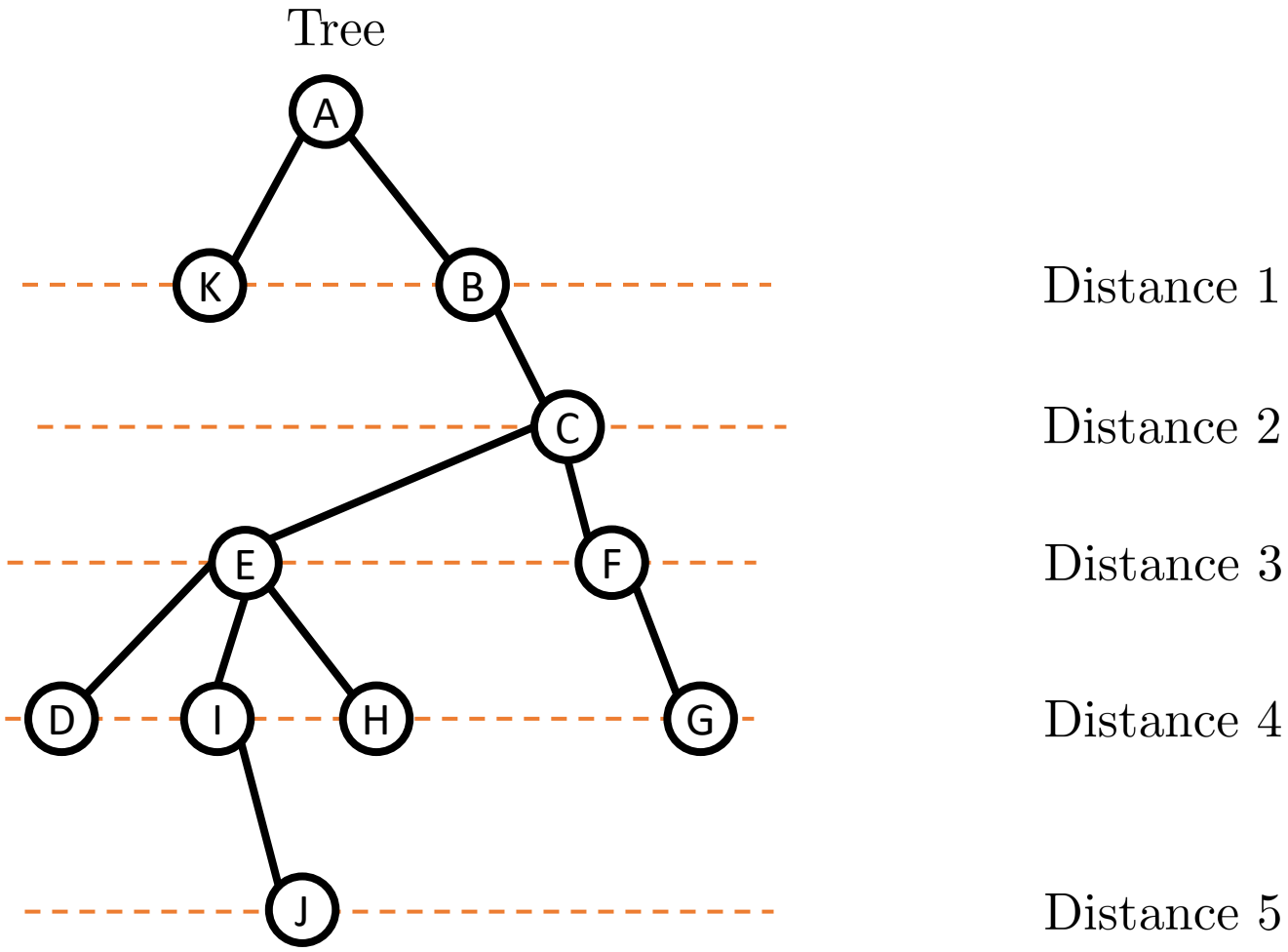
Example



Distance between A to H is 4 hops.

6.1 Local Clustering Coefficient

Note (Breadth-first search)

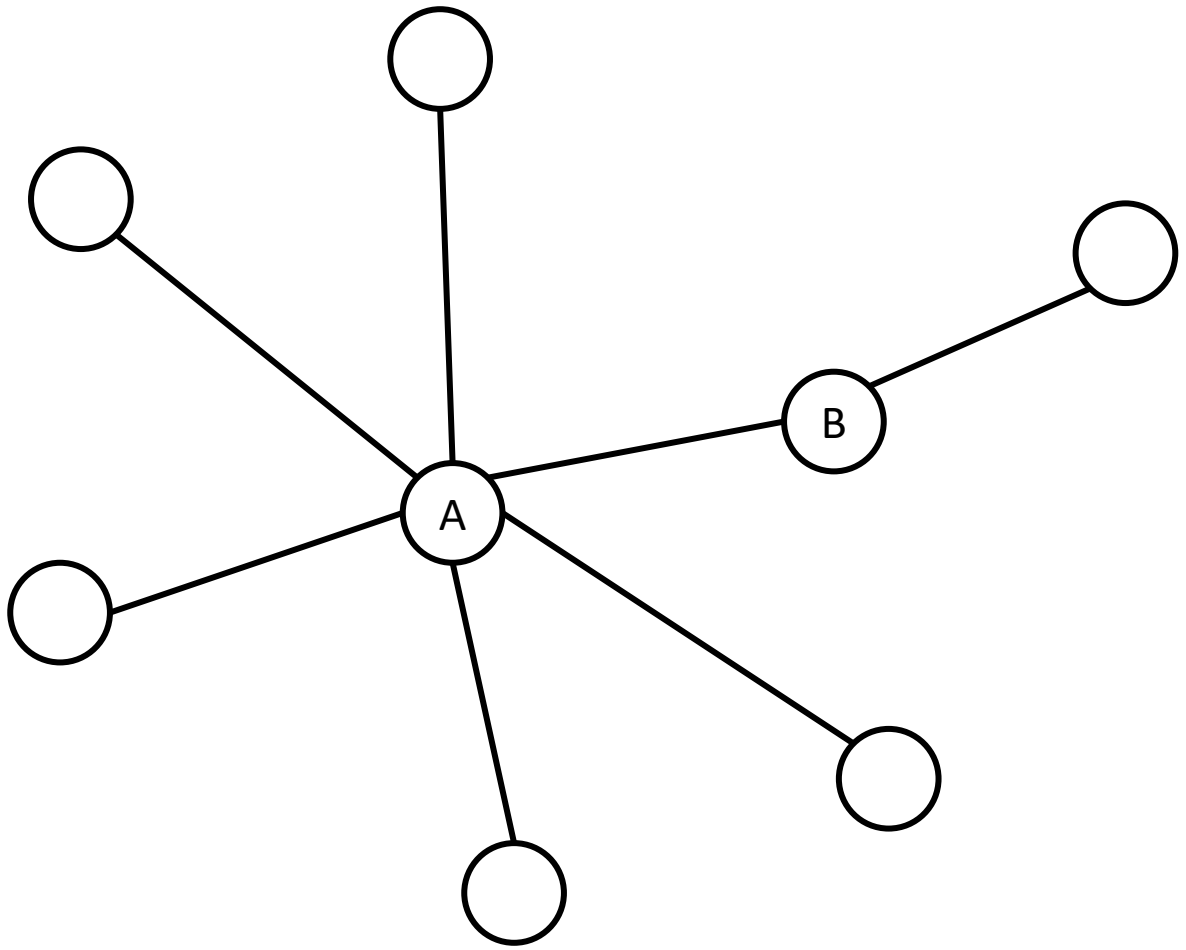


6.1 Local Clustering Coefficient

Note

- ⋆ Average distance
- ⋆ Diameter : the maximum distance
- ⋆ Eccentricity of a node n is the largest distance between n and all other nodes.
- ⋆ Radius : the minimum of eccentricity
- ⋆ Periphery is the set of nodes that have eccentricity equal tho the diameter
- ⋆ Center is the set of nodes that have eccentricity equal to the radius

6.2. Centrality: Degree

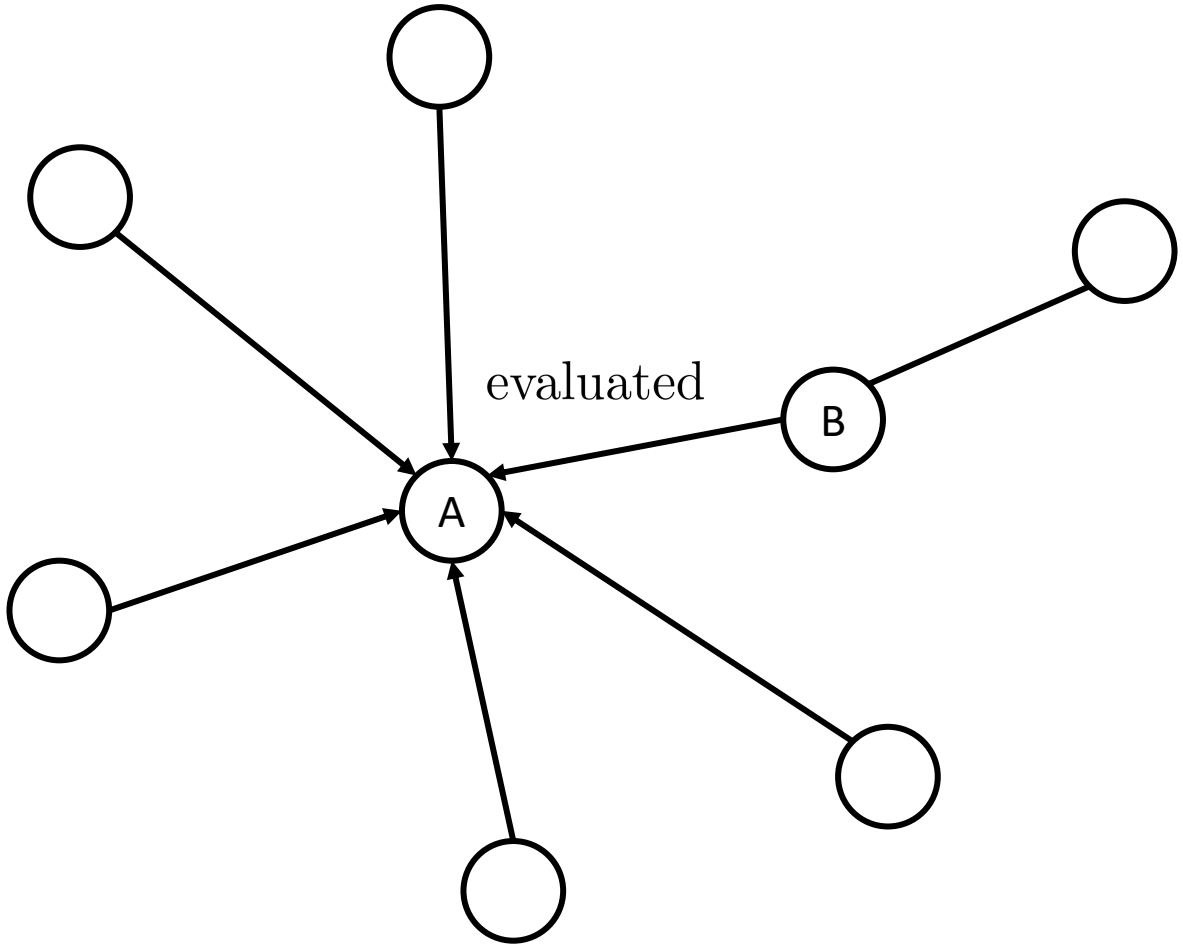


Degree of A: 6

Degree of B: 2

6.2. Centrality: Eigenvalue centrality

- give each vertex a score proportional to the sum of the scores of neighbors



6.2. Centrality: Eigenvalue centrality

Setting a score $x_i = 1$ for all i initially. Define $x_i' = \sum_j A_{ij} x_j$

$\overset{matrix}{\Longrightarrow} X' = AX,\ X(0) = (1,\ 1, \cdots, 1)^T$

Iterating this process, after t times, $X(t) = A^t X(0)$

Let $Au_i = k_i u_i$ then since u_i's are eigenvector, $[u_1,\ u_2, \cdots, u_n]$ is basis for $\mathbb{R}^n$. Then we can write $X(0) = \sum_{i=1}^{n} c_i u_i$

$$\Longrightarrow\ X(t) = A^t[\sum c_i u_i] = \sum_i c_i A^t u_i = \sum_i c_i A^{t-1}(Au_i) = \sum_i c_i A^{t-1} k_i u_i = \sum_i c_i k_i^t u_i$$

Assuming $k_1 > k_2 > \cdots > k_n > 0$, $X(t) = (k_1)^t \sum_i c_i \left[\frac{k_i}{k_1}\right]^t u_i$

Since $0 < \frac{k_i}{k_1} < 1$, $X(t) \to c_1 (k_1)^t u_1$ as $t \to \infty$.

Therefore, we can say that the centrality X satisfies $AX = k_1 X$. i.e.,

$$x_i = \frac{1}{k_1} \sum_{j=1}^{n} A_{ij} x_j$$

6.2. Centrality: Eigenvalue centrality

Setting a score $x_i = 1$ for all i initially. Define $x_i' = \sum_j A_{ij} x_j$

$\overset{matrix}{\Longrightarrow} X' = AX,\ X(0) = (1,\ 1, \cdots, 1)^T$

Iterating this process, after t times, $X(t) = A^t X(0)$

Let $Au_i = k_i u_i$ then since u_i's are eigenvector, $[u_1,\ u_2, \cdots, u_n]$ is basis for $\mathbb{R}^n$. Then we can write $X(0) = \sum_{i=1}^{n} c_i u_i$

$$\Longrightarrow\ X(t) = A^t[\sum c_i u_i] = \sum_i c_i A^t u_i = \sum_i c_i A^{t-1}(Au_i) = \sum_i c_i A^{t-1} k_i u_i = \sum_i c_i k_i^t u_i$$

Assuming $k_1 > k_2 > \cdots > k_n > 0$, $X(t) = (k_1)^t \sum_i c_i \left[\frac{k_i}{k_1}\right]^t u_i$

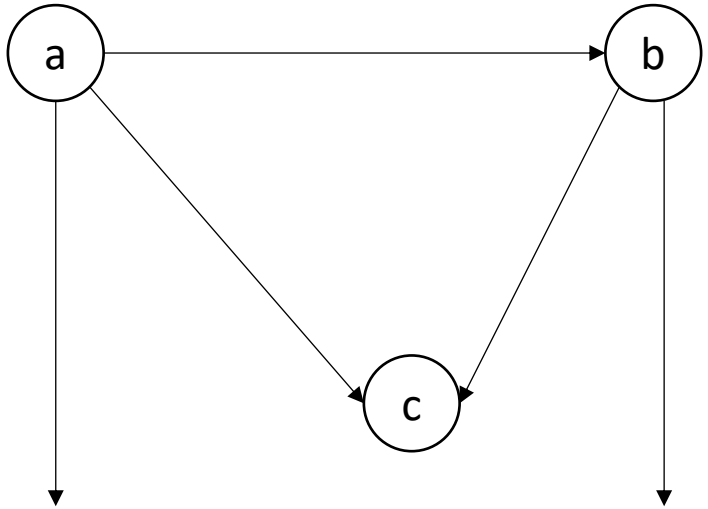
Since $0 < \frac{k_i}{k_1} < 1$, $X(t) \to c_1 (k_1)^t u_1$ as $t \to \infty$.

Therefore, we can say that the centrality X satisfies $AX = k_1 X$. i.e.,

$$x_i = \frac{1}{k_1} \sum_{j=1}^{n} A_{ij} x_j$$

6.2. Centrality: Eigenvalue centrality

Example 1.2.1



$$A_{ai} = 0 \implies X_a = 0$$

Also, $X_b = 0$ since $X_a A_{ba} = 0$ and $A_{bi} = 0$ for $i \neq a$

Therefore, Eigenvalue centrality is good for undirected network.

6.2 Centrality: Katz centrality

$x_i = \alpha \sum A_{ij} x_j + \beta$ where β is free amount$(\neq 0) \overset{matrix}{\Longrightarrow} X = \alpha A X + \beta \cdot \mathbf{1}$ for $\mathbf{1} = (1, 1, \cdots, 1)^T$

$\Longrightarrow (I - \alpha A)X = \beta \cdot \mathbf{1} \Longrightarrow X = (I - \alpha A)^{-1} \beta \cdot \mathbf{1}.$

Setting $\beta = 1$, then, $X = (I - \alpha A)^{-1} \cdot \mathbf{1}$.

Note that X exists if $(I - \alpha A)$ is invertible. i.e., $(I - \alpha A)$ is not invertible $(\det(I - \alpha A) = 0)$, X diverge.

Divided by α, $\det(A - \alpha^{-1}) = 0$ is the characteristic equation and therefore, α^{-1} is an eigenvalue of A.

Let k_i be eigenvalues of A and $k_1 > k_2 > \cdots$ then $\frac{1}{k_1} < \frac{1}{k_2} < \cdots$.

Choose $\alpha^{-1} = k_1 \Longrightarrow \alpha = \frac{1}{k_1}$

To converge X, α is chosen less than $\frac{1}{k_1}$.

6.3 Transportation Network

- Data

국가 표준노드링크(ITS):
http://nodelink.its.go.kr/Default.aspx

노드링크를 구분하는 점(링크 시작점,링크 종료점): 도로교차점,도로 시종점, 교통통제점, 도로 구조 변환점,행정구역 변환점, 도로 운영 변환점, 교통 진출입점. 그외에 ITS사업 주체가 필요에 따라 정하는 지점

- Python visualization

folium

Python Data, Leaflet.js Maps

folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Manipulate your data in Python, then visualize it in a Leaflet map via folium.

Github Link : https://github.com/python-visualization/folium

6.3 Transportation Network

- Read data and preprocessing

Read processed Data for whole Incheon

```
nodes = pd.read_csv('incheon_nodes_150105.csv')
links = pd.read_csv('incheon_links_150105.csv')
```

```
nodes = nodes[['Id','NODE_NAME','STNL_REG','latitude','longitude']]
incheon_id = dict()
for i in nodes['Id'] :
    incheon_id[i] = 0
len(nodes)
```

```
2200
```

Table for STNL_REG code

27710	대구광역시	달성군	157	28110	인천광역시	중구	161
28140	인천광역시	동구	162	28170	인천광역시	남구	163
28185	인천광역시	연수구	164	28200	인천광역시	남동구	165
28237	인천광역시	부평구	166	28245	인천광역시	계양구	167
28260	인천광역시	서구	168	28710	인천광역시	강화군	169
28720	인천광역시	옹진군	170	29110	광주광역시	동구	175

```
source_in = links['Source'].apply(lambda x : x in incheon_id) # check Sourc
target_in = links['Target'].apply(lambda x : x in incheon_id) # check Targe
# source_in and target_in are boolean type pandas.Series which contains Tru
```

```
incheon_links = links[source_in & target_in] # contain if both target and s
incheon_links.head()
```

	Source	Target
0	1610003100	1680004000
1	1610003100	1610003000
2	1680003800	1610003100
3	1610000800	1610000700
4	1610008600	1630014900

6.3 Transportation Network

- Network Construction

```
G = nx.Graph()
# R is the Earth's radius
R = 6371e3

for idx,row in nodes.iterrows():
    G.add_node(row['Id'],Label=row['NODE_NAME'],latitude=row['latitude'], 
for idx,row in incheon_links.iterrows():
    ## Calculate the distance between Source and Target Nodes
    lon1 = float(nodes[nodes['Id'] == row['Source']]['longitude'] * np.pi/
    lat1 = float(nodes[nodes['Id'] == row['Source']]['latitude'] * np.pi/18
    lon2 = float(nodes[nodes['Id'] == row['Target']]['longitude'] * np.pi/
    lat2 = float(nodes[nodes['Id'] == row['Target']]['latitude'] * np.pi/18

    d_lat = lat2 - lat1
    d_lon = lon2 - lon1
    a = np.sin(d_lat/2) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(d_lon/
    c = 2 * np.arctan2(a**0.5, (1-a) ** 0.5)
    d = R * c
    ## Link attribute, 'Source', 'Target' and weight = 'Length between ther
    G.add_edge(row['Source'],row['Target'],weight = d)
```

```
print(nx.info(G))
```

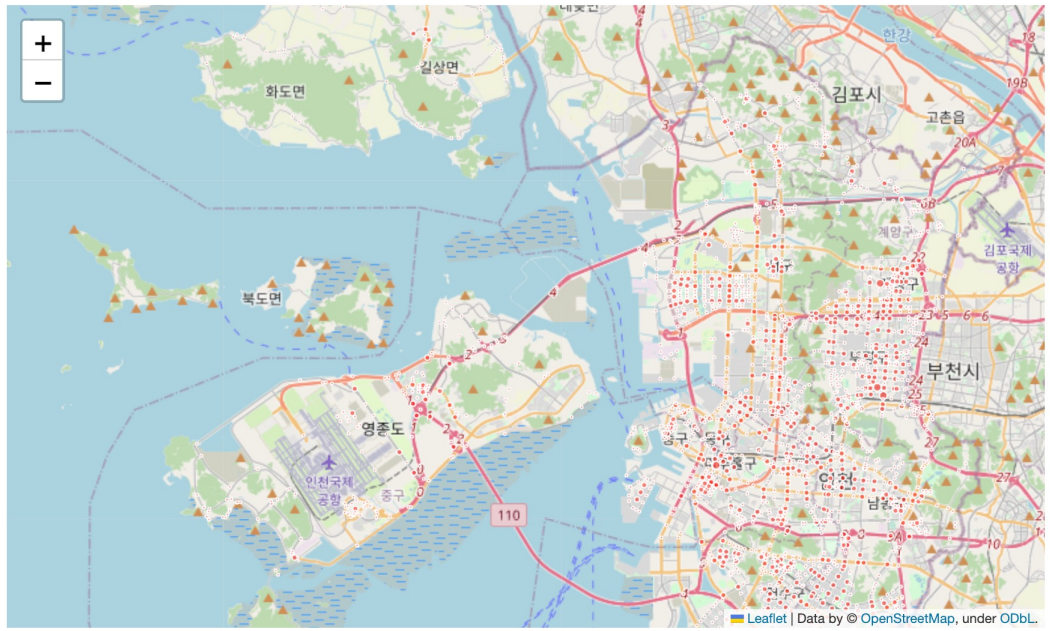
```
Name:
Type: Graph
Number of nodes: 2200
Number of edges: 3233
Average degree:   2.9391
```

6.3 Transportation Network

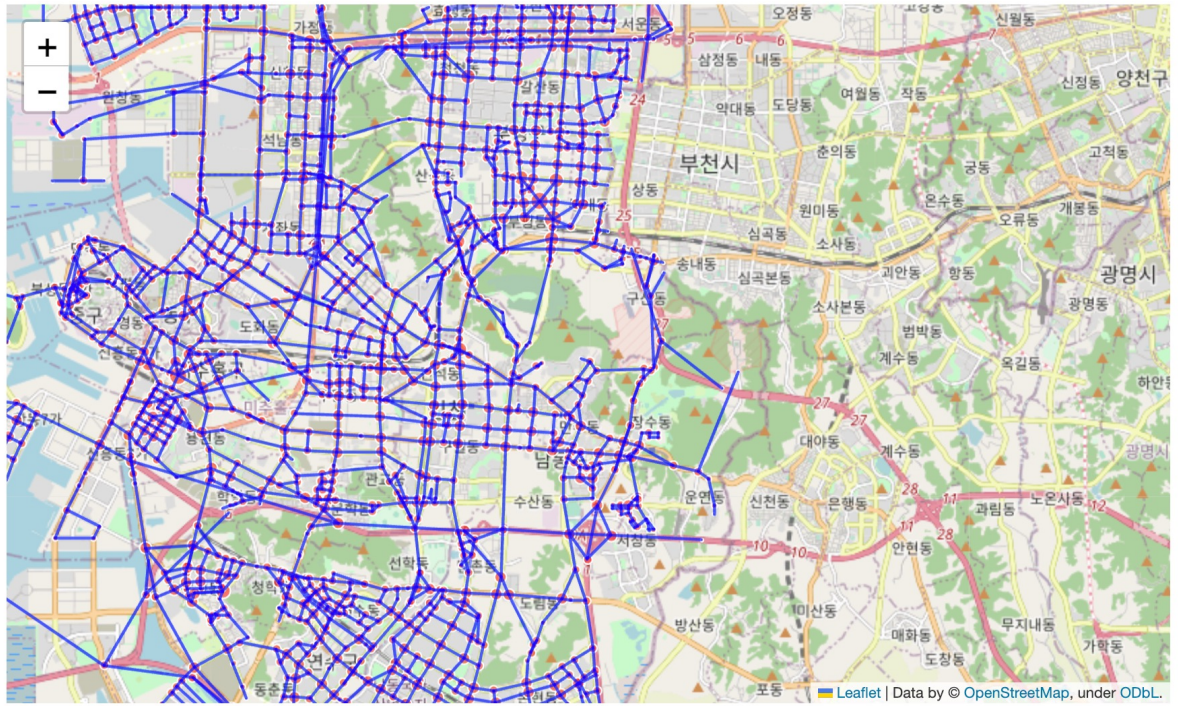
- Visualization of Network

```
map_osm = folium.Map(location=std_point, zoom_start=10)

for ix, row in nodes.iterrows():
    location = (row['latitude'], row['longitude'])
    folium.Circle(
        location=location,
        radius=incheon_id[row['Id']] * 15,
        color='white',
        weight=1,
        fill_opacity=0.6,
        opacity=1,
        fill_color='red',
        fill=True,  # gets overridden by fill_color
        # popup=str(row['Id'])
    ).add_to(map_osm)
    # folium.Marker(location, popup=row['NODE_NAME']).add_to(map_osm)
# it takes some time.....
map_osm
```




```
kw = {'opacity': 0.5, 'weight': 2}
for ix, row in incheon_links.iterrows():
    start = tuple(nodes[nodes['Id']==row['Source']][['latitude','longitude']].iloc[0])
    end = tuple(nodes[nodes['Id']==row['Target']][['latitude','longitude']].iloc[0])
    folium.PolyLine(
        locations=[start, end],
        color='blue',
        line_cap='round',
        **kw,
    ).add_to(map_osm)
# it takes some time.....
map_osm
```



6.3 Transportation Network

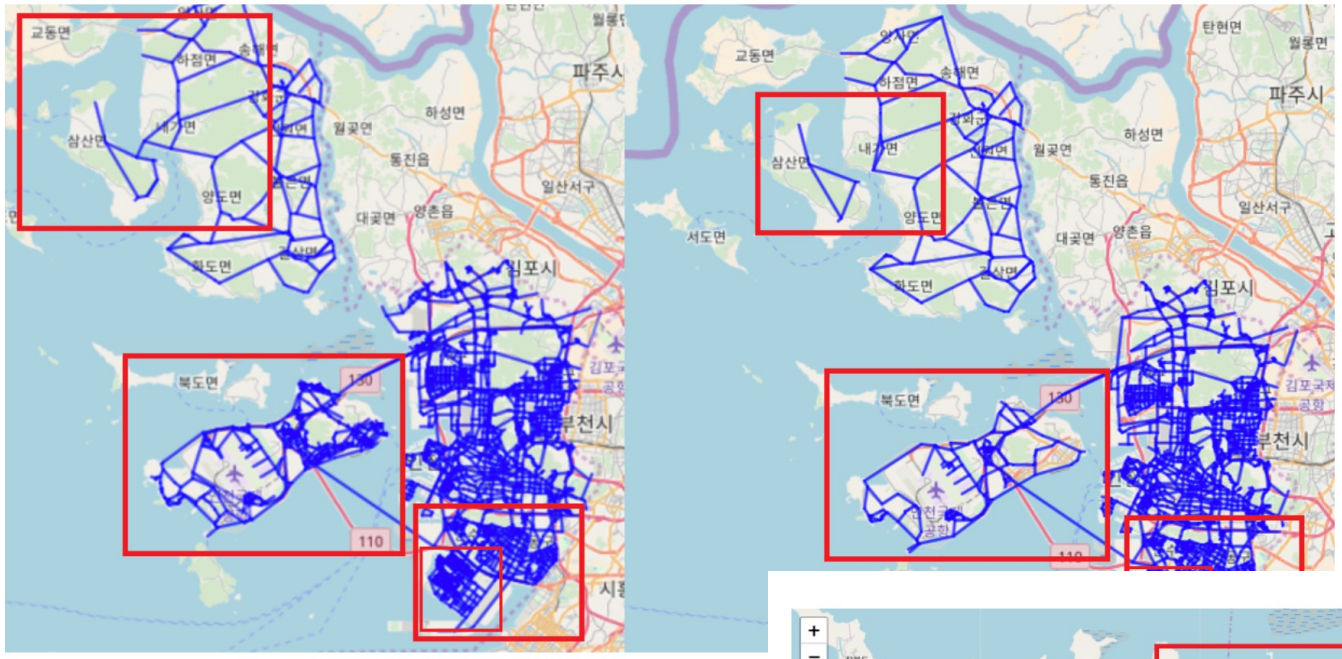


Figure. 1. Change of structural prope
Left : 2018.02.05, Right : 20

- Analysis of Network

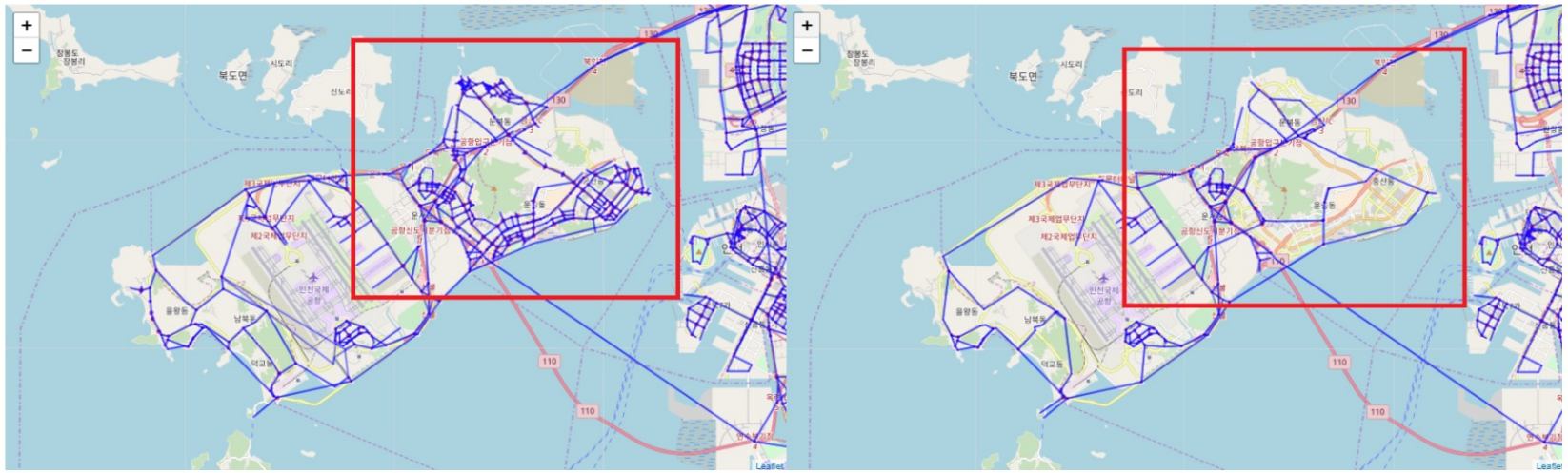
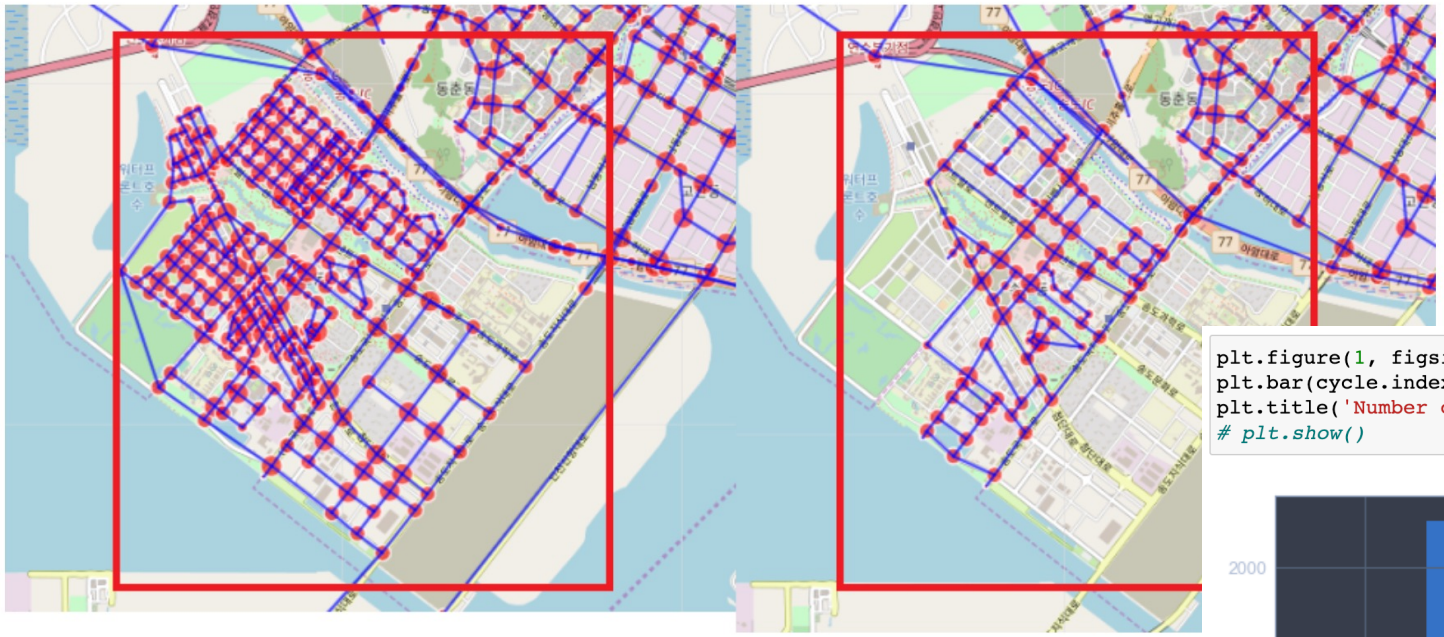


Figure. 1.2. Change of structural property of Yeongjong district.
Left : 2018.02.05, Right : 2015.01.05

6.3 Transportation Network

Songdo International City(송도신도시) was also affected by this project.

- Below figure shows the change of structural property of the Songdo International City.



- Analysis of Network

```
plt.figure(1, figsize=(14, 7))# control figure size
plt.bar(cycle.index, cycle)
plt.title('Number of Length r Cycle, 2015', fontsize=14);
# plt.show()
```