Lecture 9

Paper: The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles

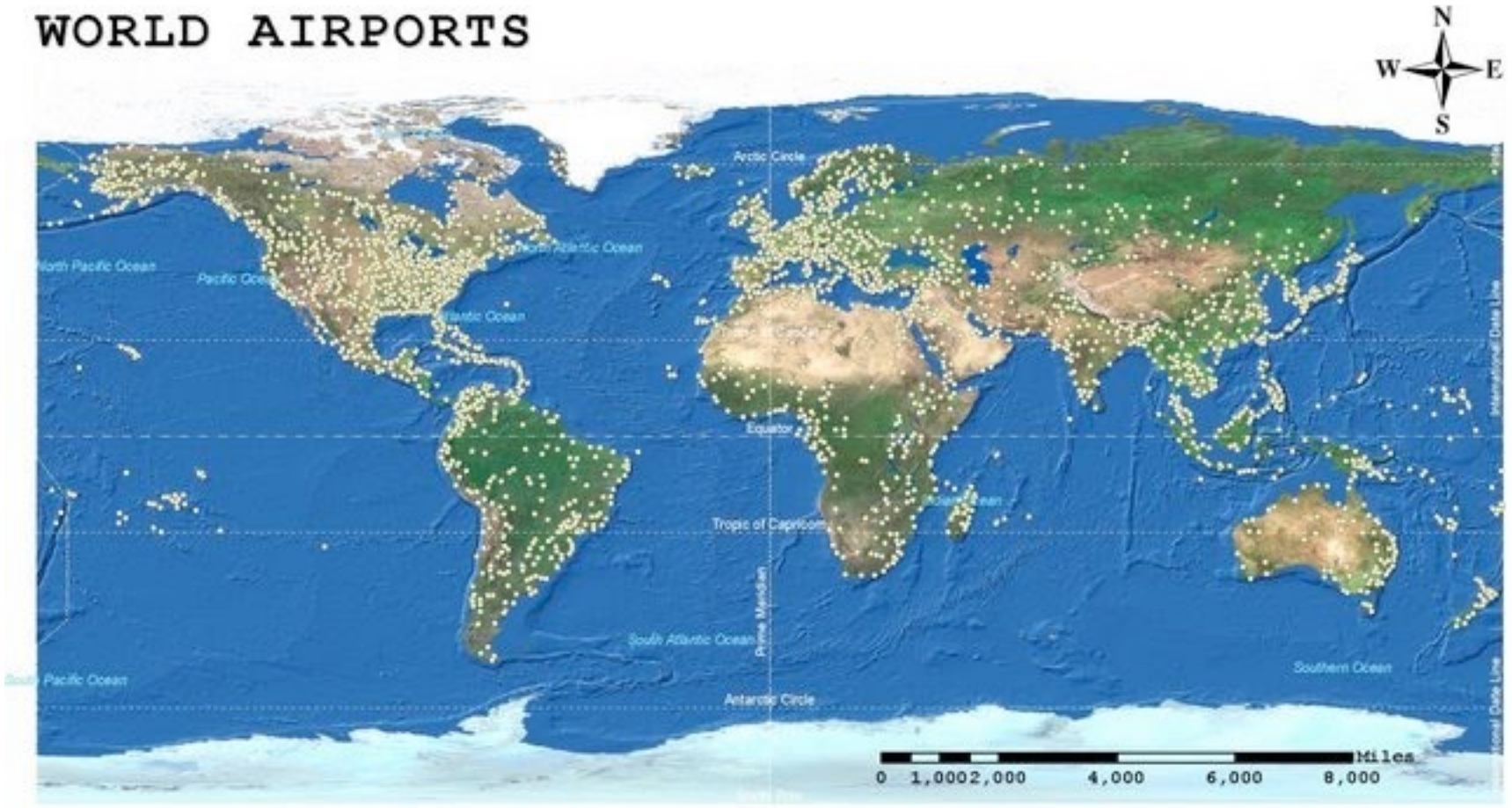
Method: Community Detection in Networks (Network Partition in Clusters)

References

- The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles.Guimera, R, Mossa, S, Turtschi, A, Amaral, LAN.

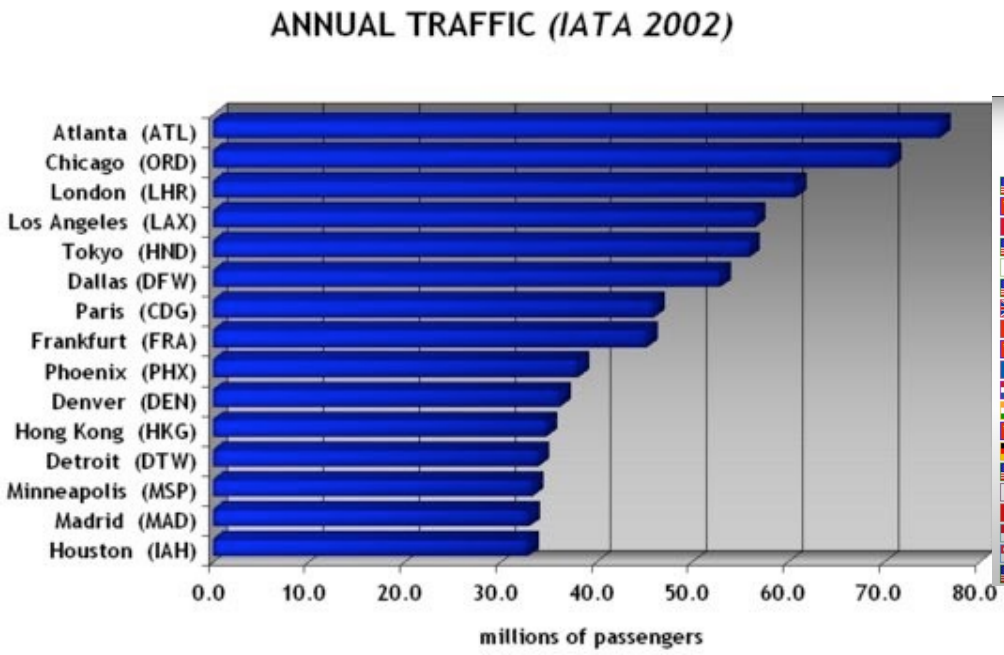
  Proc. Natl. Acad. Sci. U. S. A. 102, 7794-7799 (2005)

- Finding and evaluating community structure in networks, M. E. J. Newman and M. Girvan, Phys. Rev. E 69, 026113 (2004).

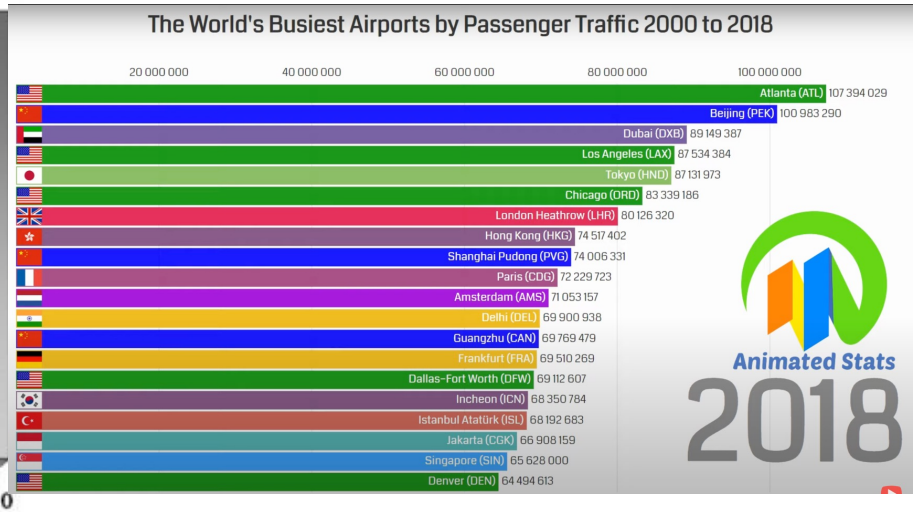


41,700 airports all over the world according to the Central Intelligence Agency. The United States alone has over 13,000 airports listed with the Central intelligence Agency. Coming in second is the country of Brazil with a little over 4,000 airports and from there the numbers drop drastically per country

The Global Airport Database (GADB) is a **FREE** downloadable database of **9300** airports big and small from all around the world.
https://www.partow.net/miscellaneous/airportdatabase/



Annual Traffic data 2018



https://www.youtube.com/watch?v=Te4JAKVCNbo

Impact in economies

Disease Spreading (SARS, influenza, COVID19)

A metric of influential spreading during contagion dynamics through the air transportation network. C. Nicolaides, L. Cueto-Felgueroso, M. C. Gonzalez and R. Juanes, PLoS ONE, 7(7), e40961 (2012)
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0040961

Passangers per year world wide?

700 million passengers each year

Percentage of US flights in/out Chicago?

Chicago (10% of the total commercial flights)

How many?

~2,700 daily

blue-sky day ----- landing rate 100 per hour

low-clouds days ----- landing rate 72 landing per hour

Flight delay costs per year --- 150 to 200 billion Euro

Much research has been done in optimal network design

System level analysis >> Network Analysis

3,883 locations (villages towns and cities) (OAG max database)
Nov. 1, 2000, to Oct 31, 2001

>800 of the world's airlines

531,574 unique non-stop passenger flights

27,051 city pairs have nonstop connections

Symmetric network

https://www.oag.com/

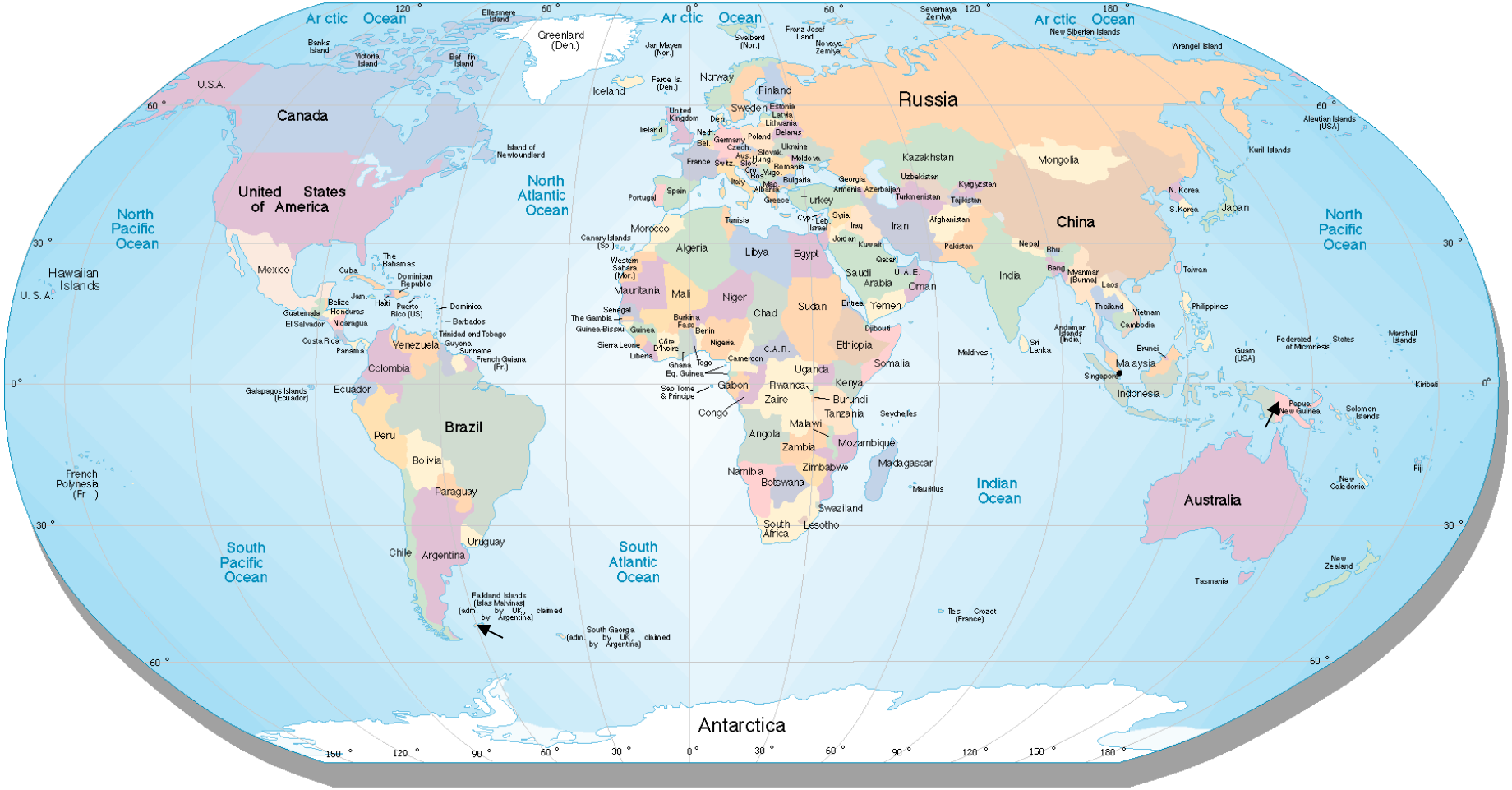
Giant component 3,663 with d=4.4

56% of the cities are connected by 4 steps or less

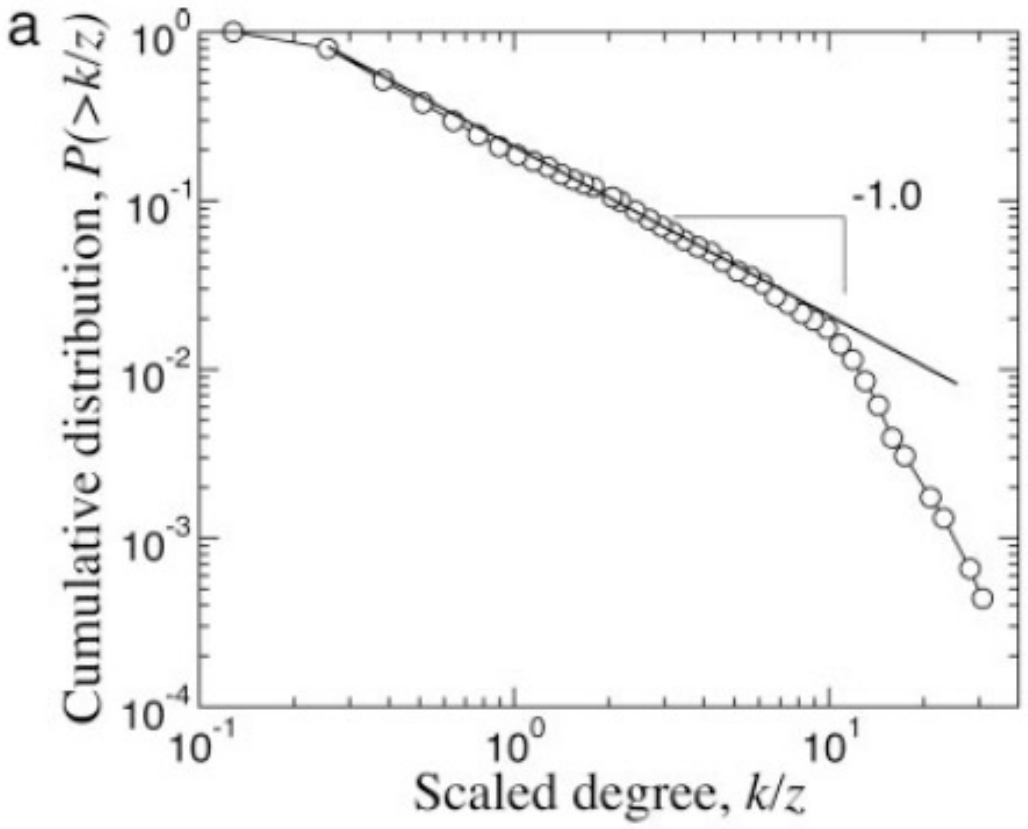
C=0.62, whereas its randomized version C=0.049

Worldwide air transportation network is Small World

The farthest airports are Mount Pleasant in the Falkand Islands And Wasu, Papau, New Guinea (15 diffent flights)



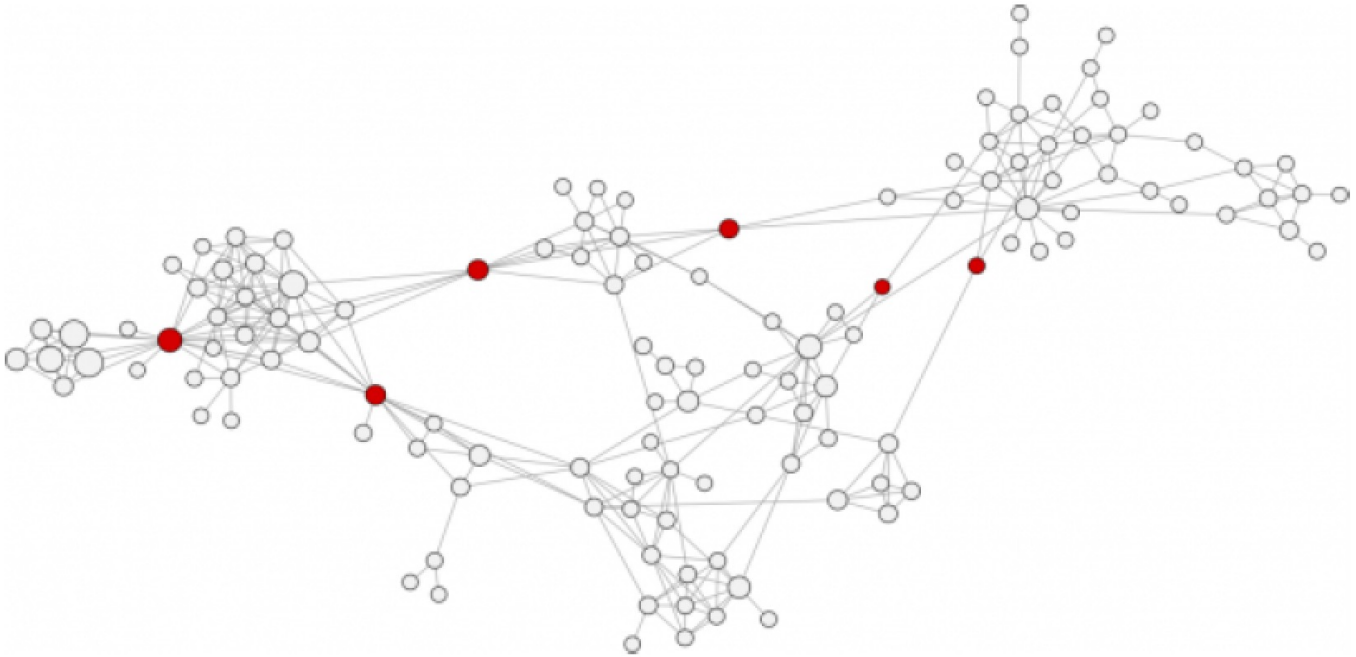
Cummulative Degree distribution



$$P(>k) \propto k^{-\alpha} f(k/k_{\times}) \qquad \alpha = 1.0 \pm 0.1$$

Amaral, L. A. N., Scala, A., Barthelemy, M. & Stanley, H. E. (2000) Proc. Natl. Acad. Sci. USA 97, 11149–11152.

Betweenness centrality



If Betweenness Centrality is high a node bridges many shortest paths

Betweenness

Idea: a node is the more central the *more often it is crossed by paths*

$$b_i = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}}$$

$\sigma_{hj} =$ number of shortest paths from h to j

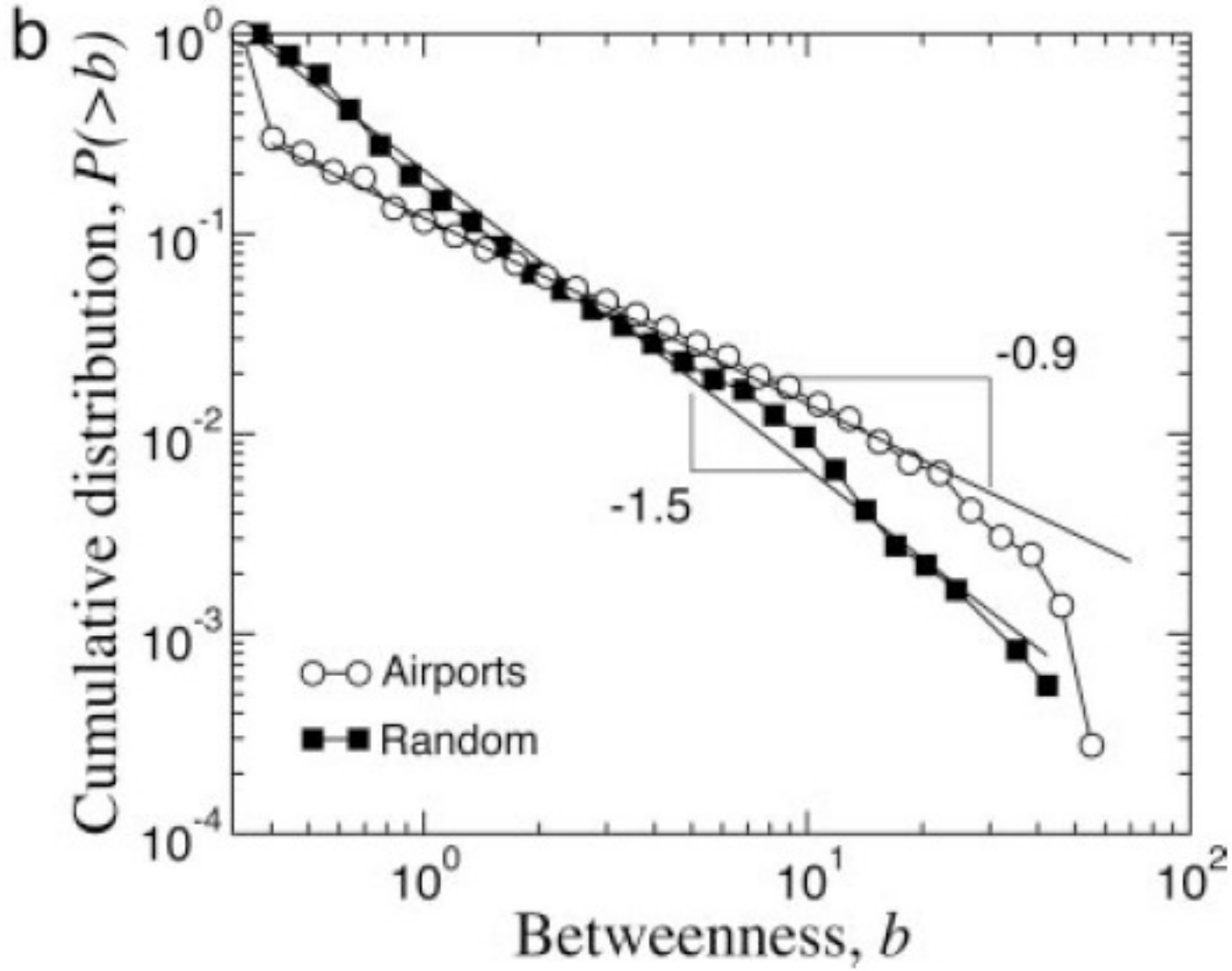
$\sigma_{hj}(i) =$ number of shortest paths from h to j running through i

Betweenness

- Betweenness can be easily extended to links
- **Link betweenness**: fraction of shortest paths among all possible node pairs that pass through the link

```
nx.betweenness_centrality(G)       # dict nodes ->
                                   # betweenness centrality
nx.edge_betweenness_centrality(G) # dict links ->
                                   # betweenness centrality
```

Note: This is important for Road Network Analysis

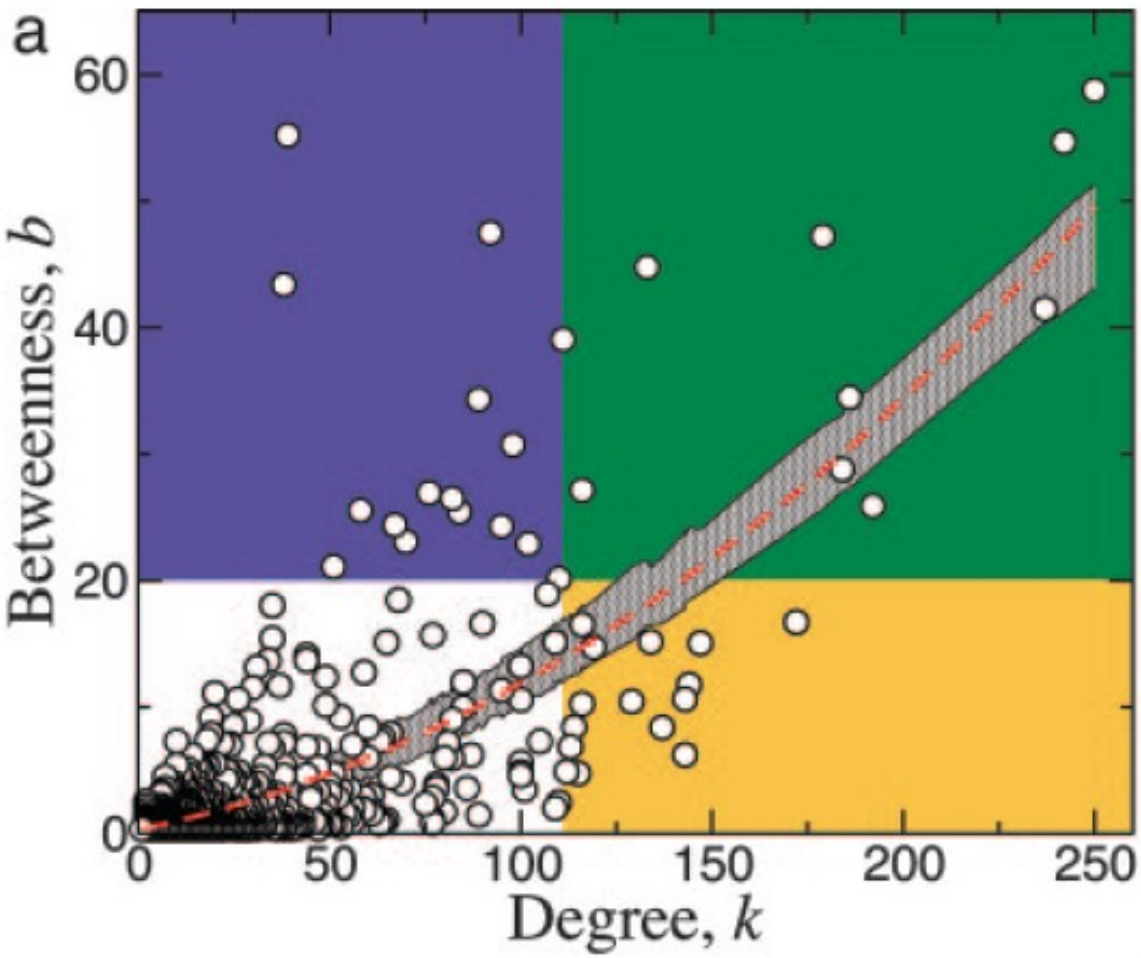


$$P(>b) \propto b^{-\nu} g(b/b_{\times})$$

$$\nu = 0.9 \pm 0.1$$

***bi*= *Bi*/<B>, where <B> represents theaverage betweenness for the network.**

Are the most connected cities also the most central?



Most Connected Cities

Most Central Cities

Table 2. The 25 most central cities in the worldwide air transportation network

Rank	City	b	b/b_{ran}	Degree
1	Paris	58.8	1.2	250
2	Anchorage*	55.2	16.7	39
3	London	54.7	1.2	242
4	Singapore*	47.5	4.3	92
5	New York	47.2	1.6	179
6	Los Angeles	44.8	2.3	133
7	Port Moresby*	43.4	13.6	38
8	Frankfurt	41.5	0.9	237
9	Tokyo	39.1	2.7	111
10	Moscow	34.5	1.1	186
11	Seattle*	34.3	3.3	89
12	Hong Kong*	30.8	2.6	98
13	Chicago	28.8	1.0	184
14	Toronto	27.1	1.8	116
15	Buenos Aires*	26.9	3.2	76
16	São Paulo*	26.5	2.8	82
17	Amsterdam	25.9	0.8	192
18	Melbourne*	25.5	4.5	58
19	Johannesburg*	25.4	2.6	84
20	Manila*	24.4	3.5	67
21	Seoul*	24.3	2.1	95
22	Sydney*	23.1	3.2	70
23	Bangkok*	22.9	1.8	102
24	Honolulu*	21.1	4.4	51
25	Miami*	20.1	1.4	110

Cities are ordered according to their normalized betweenness. We also show the ratio of the actual betweenness of the cities to the betweenness that they have after randomizing the network.

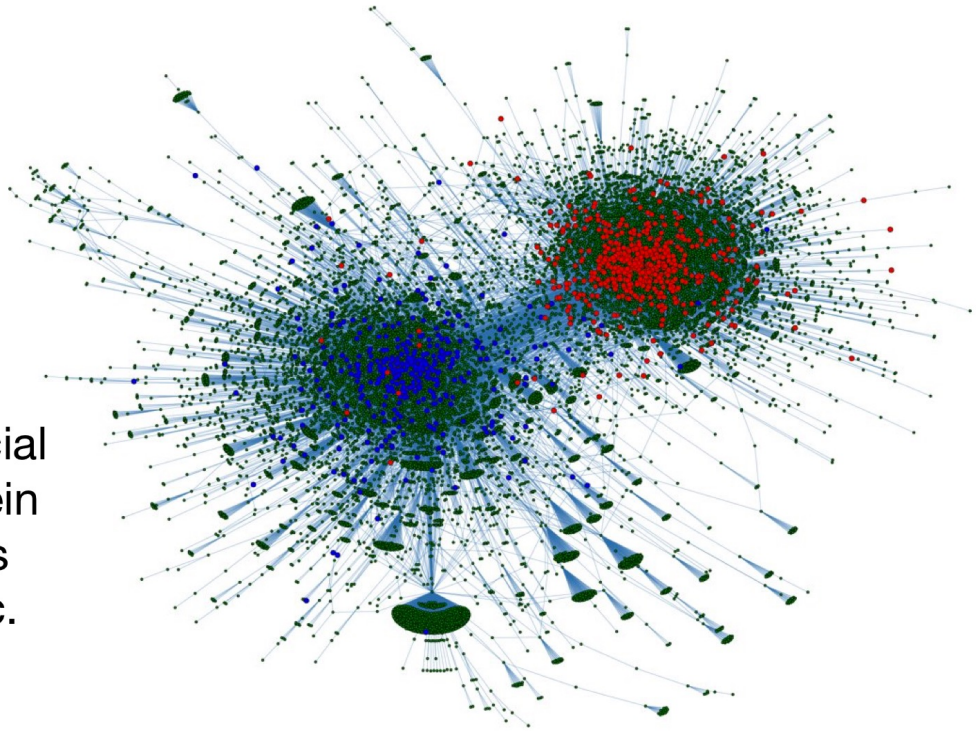
*These cities are not among the 25 most connected.

Table 1. Number of locations with airports by major geographic region

Region	No. of locations
Africa	364
Asia and Middle East	719
Europe	691
Latin America	523
North America	1,064
Oceania	522

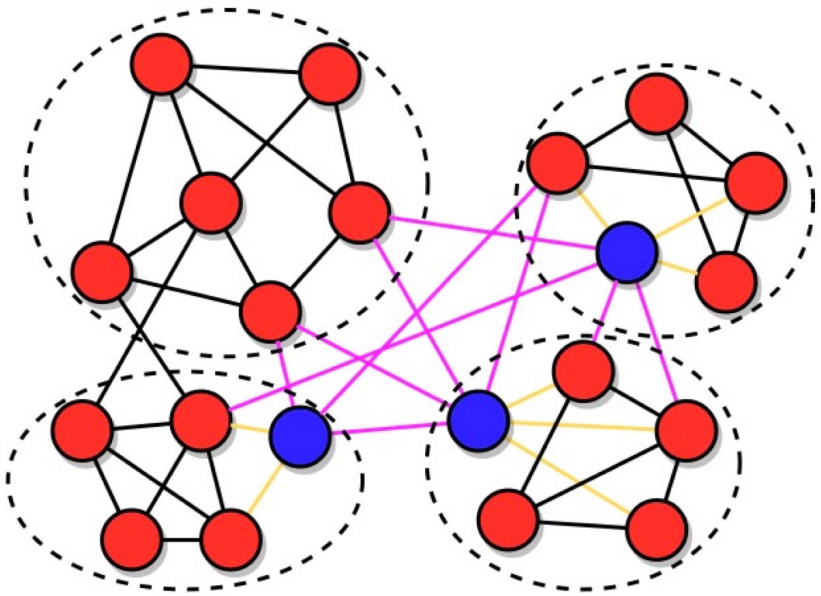
Community structure

- **Example:** Twitter users with strong political preferences tend to follow those aligned with them and not to follow users with different political orientation
- **Other examples:** social circles in social networks, functional modules in protein interaction networks, groups of pages about the same topic on the Web, etc.



Community definitions based on cohesion versus separation

- **Less stringent definitions** of strong and weak community:
  - **Strong community:** subnetwork such that each node has more neighbors inside it than in any other community
  - **Weak community:** subnetwork such that the sum of the internal degrees of its nodes exceeds the total number of neighbors that the nodes have in any other community



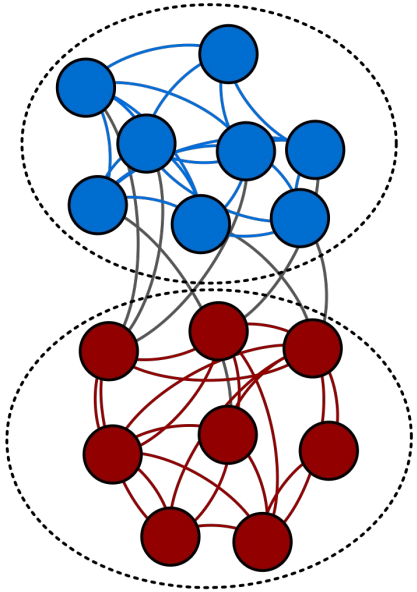
Modularity

$$Q = \frac{1}{L} \sum_{C} \left(L_C - \frac{k_C^2}{4L} \right)$$

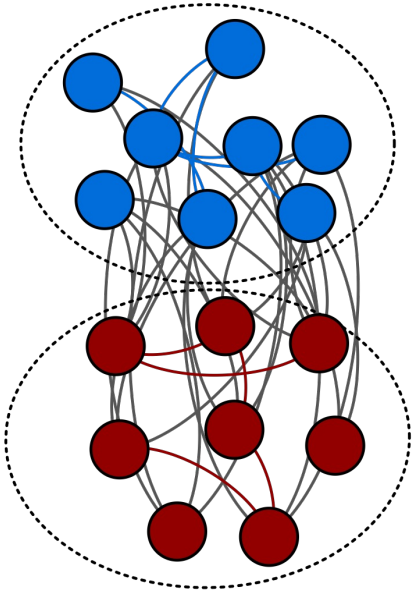
- L = number of links in the network
- L_C = number of internal links in community C
- k_C = degree of community C
- $k^2_C/4L$ = expected number of internal links in community C

Modularity

- Let's explain the origin of $k^2_C / 4L$
- Random links are formed by matching pairs of **stubs** (half-links) chosen at random
- The total number of stubs attached to C is k_C
- The probability to select one of those stubs at random is $k_C/2L$ because $2L$ is the total number of stubs of the network (each link yields two stubs)



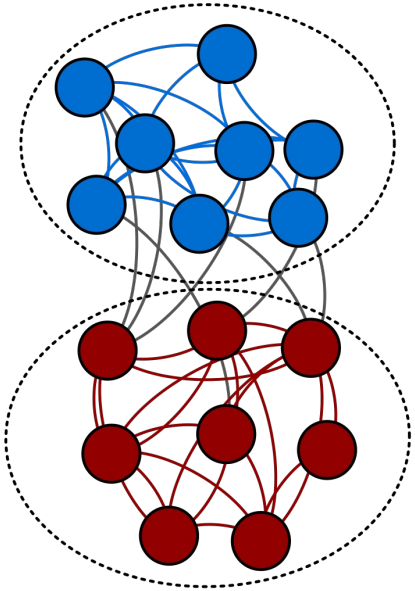
Original network



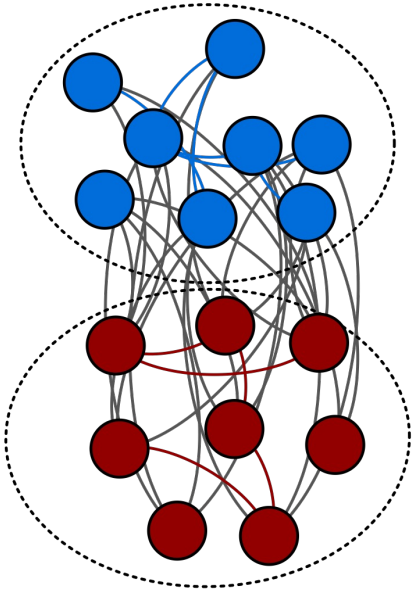
Randomized network

Modularity

- For a random link to connect two nodes in the same cluster *C*, two stubs must be selected from *C*
- The probability to pick two stubs from *C* at random is (roughly) the product of the probabilities of selecting each one: $p_C = (k_C/2L) \cdot (k_C/2L) = k^2_C / 4L^2$
- Since there are *L* links in the network, and each has a probability p_C to end up within *C*, the expected number of internal links in *C* is $k^2_C / 4L$

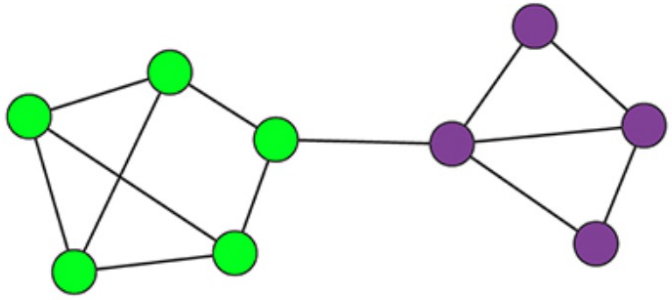


Original network



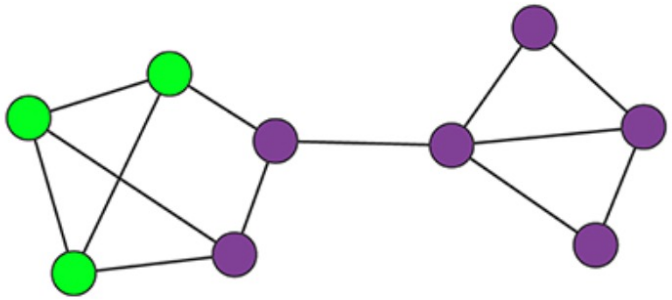
Randomized network

a. OPTIMAL PARTITION



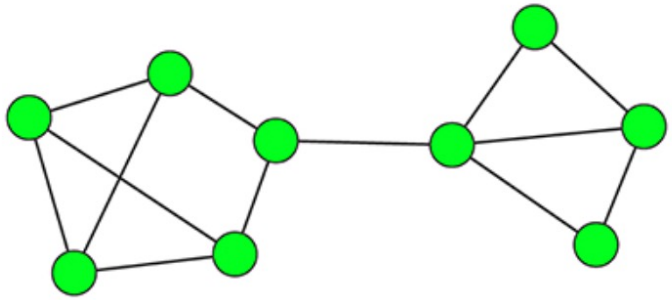
1/13[(7-14^2/(4*13))+(5-10^2/(4*13)) = **0.48**

b. SUBOPTIMAL PARTITION



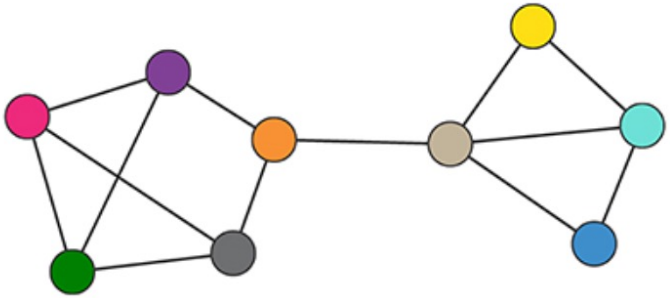
1/13[(3-7^2/(4*13))+(7-(1+2+4+2x2+3)^2/(4*13))] = **0.41**

c. SINGLE COMMUNITY



1 - (5x3+4+3+2x2)^2/(4x13^2)=1-26^2/(676)= **0**

d. NEGATIVE MODULARITY



(5x3^2+4^2+3^2+2x2^2)/(4x13^2)=78/(676)= **-0.12**

$$Q = \frac{1}{L} \sum_{C} \left(L_C - \frac{k_C^2}{4L} \right)$$

Modularity: features

- *Q < 1 for every partition of any network*
- *Q = 0 for the partition in which the whole graph is one community*
- *Q can be negative (e.g., partition in N groups of one node each)*
- *For most networks, Q has a non-trivial maximum between 0 and 1*

```
# returns the modularity of the input partition
modularity = nx.community.quality.modularity(G,partition)
```

Louvain algorithm

Procedure:

1. Start: each node is assigned to a different community
2. Loop over the nodes: each node is put in the community of the neighbor that yields the largest modularity increase ΔQ with respect to the current partition. All nodes are revisited over and over until it is no longer possible to increase Q by moving a node to a different community
3. The network is transformed into a weighted supernetwork, where each community in the partition from step 2 is replaced by a supernode, links between supernodes are weighted by the number of links joining nodes in the corresponding groups, and links joining nodes in the same community are represented as a self-loop from the corresponding supernode to itself, with weight equal to the number of internal links
4. The procedure stops when no further grouping of the clusters in the current partition increases the modularity

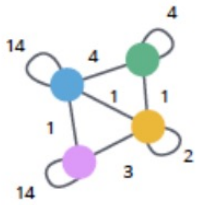
Pass 1

Step 0

Choose a start node and calculate the change in modularity that would occur if that node joins and forms a community with each of its immediate neighbors.

Step 1

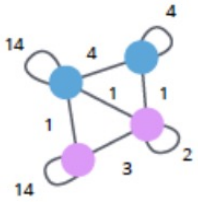
The start node joins the node with the highest modularity change. The process is repeated for each node with the above communities formed.



Step 2

Communities are aggregated to create super communities and the relationships between these super nodes are weighted as a sum of previous links. (Self-loops represent the previous relationships now hidden in the super node.)

Pass 2



Step 1

Step 2

Steps 1 and 2 repeat in passes until there is no further increase in modularity or a set number of iterations have occurred.

https://sites.google.com/site/findcommunities/home?authuser=0

Louvain algorithm

Limits:

- Like Newman's algorithm, it is a greedy method, in that it tries to find the best modularity partition at each level of agglomeration. Therefore it yields solutions with sub-optimal modularity
- The final partition depends on the order in which nodes are visited

Big plus:

- The algorithm is very fast because, after the first iteration, the successive transformations shrink the network very quickly and typically only a handful of partitions are generated. It can detect communities in networks with millions of nodes and links

Louvain algorithm

There currently is no function in NetworkX for the Louvain algorithm. However, it is possible to implement the algorithm using the `community` module

! pip install python-louvain

```
import community

# returns the partition with largest modularity
partition_dict = community.best_partition(G)
```

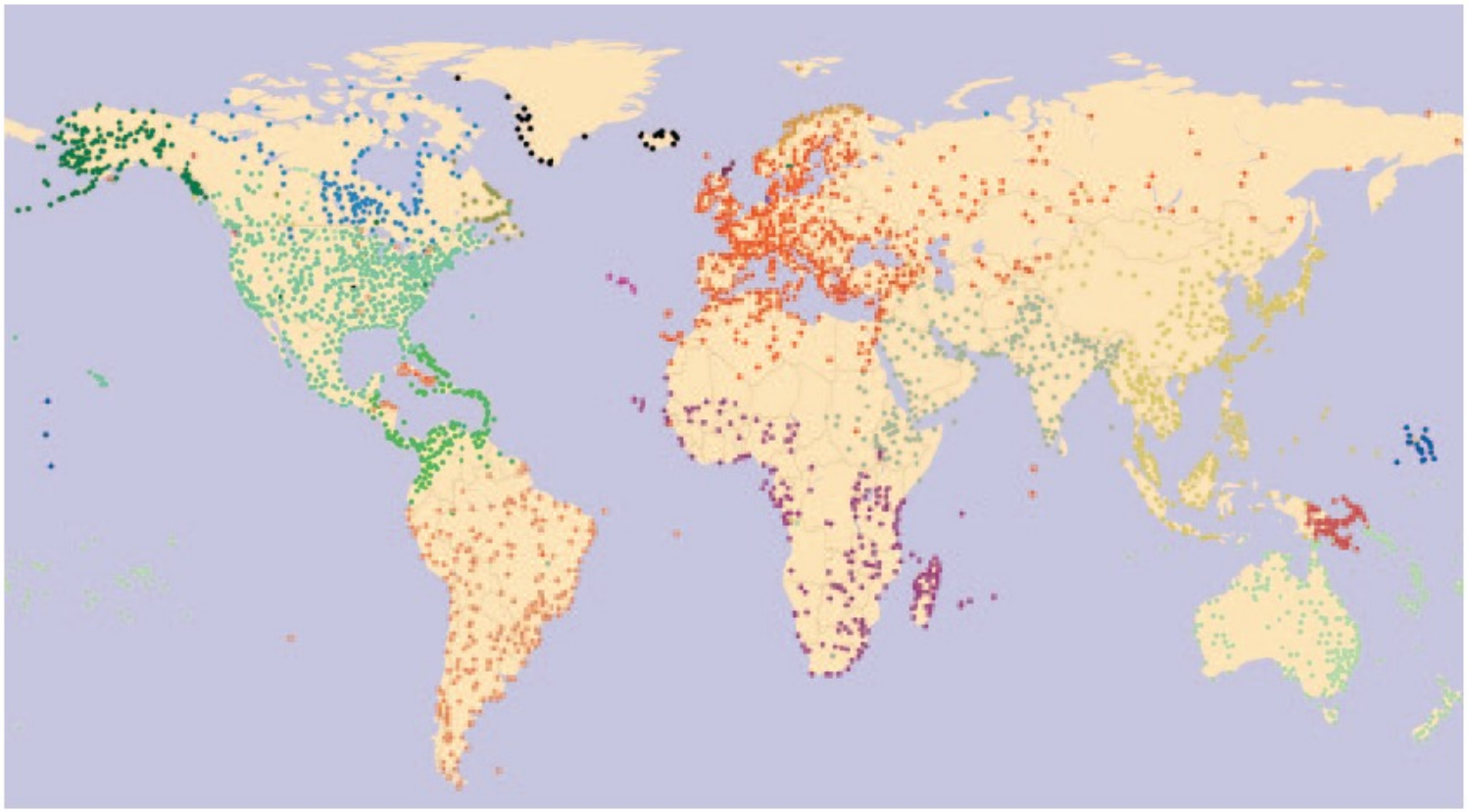


Fig. 3. Communities in the giant component of the worldwide air transportation network. Each node represents a location, and each color corresponds to a community.

Z-scores

$$z_i = \frac{\kappa_i - \bar{\kappa}_{s_i}}{\sigma_{\kappa_{s_i}}},$$

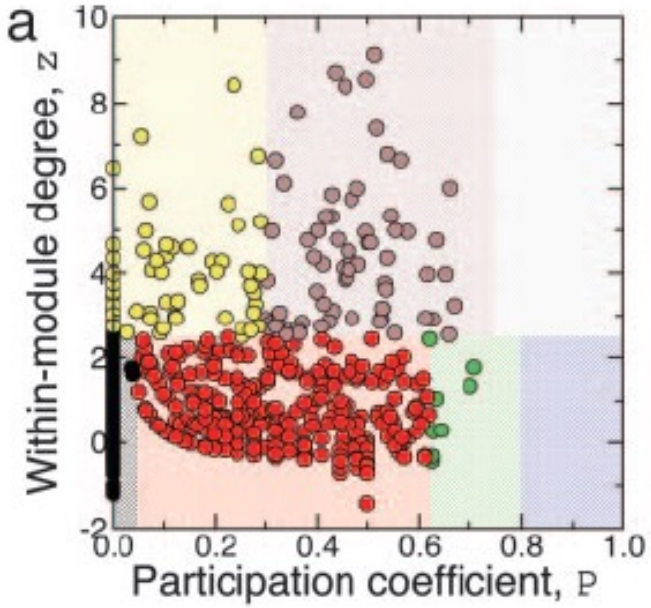
the within community degree of a node minus the average divided by the standard deviation

κ_i is the number of links of node i into community

Participation

$$P_i = 1 - \left(\frac{\kappa_{is}}{k_i}\right)^2,$$

The participation coefficient of a node is therefore close to one if its links are uniformly distributed among all of the communities and zero if all its links are within its own community.



$$z_i = \frac{\kappa_i - \bar{\kappa}_{s_i}}{\sigma_{\kappa_{s_i}}},$$

$$P_i = 1 - \sum_{s=1}^{N_M} \left(\frac{\kappa_{is}}{k_i} \right)^2,$$

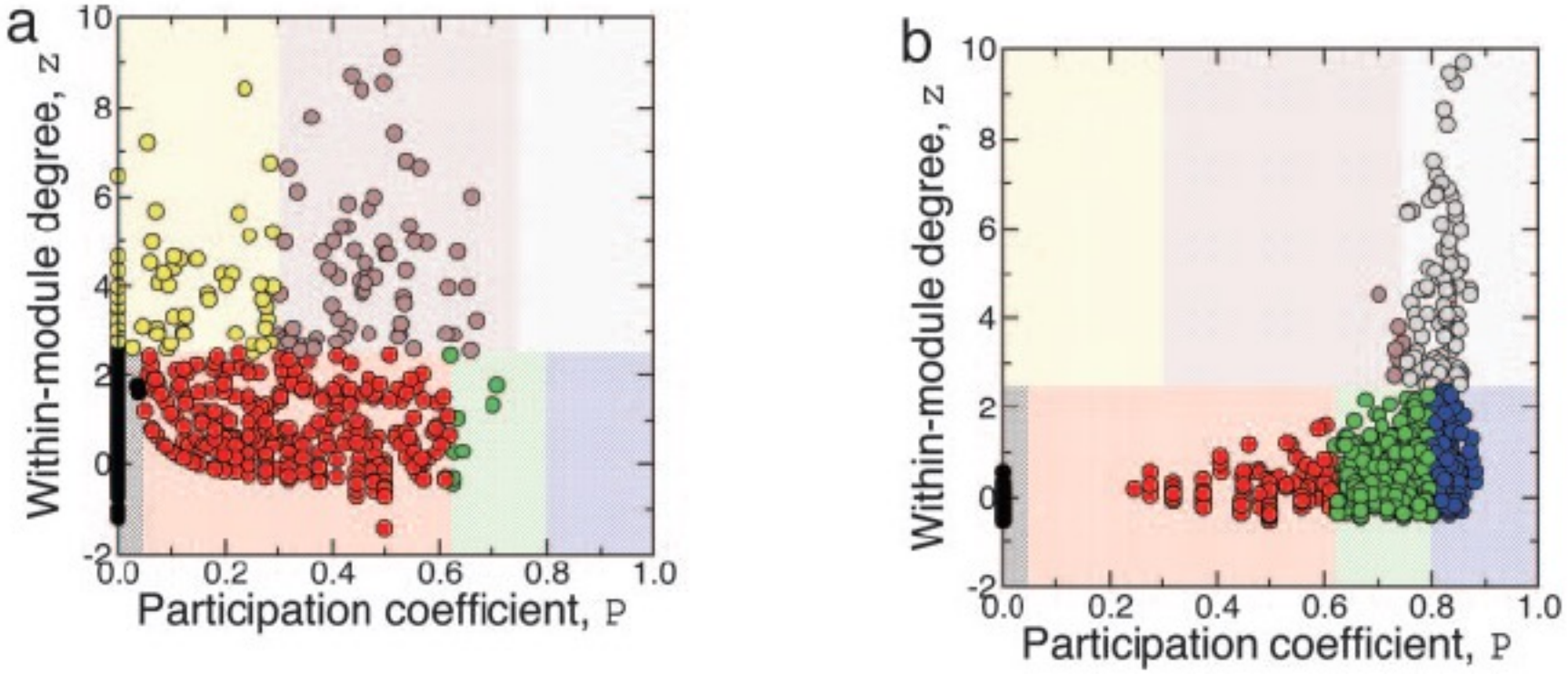
We divided nonhub nodes into four different roles as follows: (R1) "ultraperipheral nodes," i.e., nodes with all their links within their module ($P \leq 0.05$); (R2) "peripheral nodes," i.e., nodes with most links within their module ($0.05 < P \leq 0.62$); (R3) "nonhub connector nodes," i.e., nodes with many links to other modules ($0.62 < P \leq 0.80$); and (R4) "nonhub kinless nodes," i.e., nodes with links homogeneously distributed among all modules ($P > 0.80$).

R1 (black)
R2 (red)
R3 (green)

We divided hub nodes into three different roles as follows: (R5) "provincial hubs," i.e., hub nodes with the vast majority of links within their module ($P \leq 0.30$); (R6) "connector hubs," i.e., hubs with many links to most of the other modules ($0.30 < P \leq 0.75$); and (R7) "kinless hubs," i.e., hubs with links homogeneously distributed among all modules ($P > 0.75$).

R5 (yellow)
R6 (green)
R7 (white)

Randomized Network Completely different



Nonhub connectors (green), provincial hubs (yellow), and connector hubs (brown) in the worldwide air transportation network

double_edge_swap

double_edge_swap(*G, nswap=1, max_tries=100, seed=None*) [source]

Swap two edges in the graph while keeping the node degrees fixed.

A double-edge swap removes two randomly chosen edges u-v and x-y and creates the new edges u-x and v-y:

```
u--v            u  v
       becomes  |  |
x--y            x  y
```

If either the edge u-x or v-y already exist no swap is performed and another attempt is made to find a suitable edge pair.

Parameters:

G : *graph*

An undirected graph

nswap : *integer (optional, default=1)*

Number of double-edge swaps to perform

max_tries : *integer (optional)*

Maximum number of attempts to swap edges

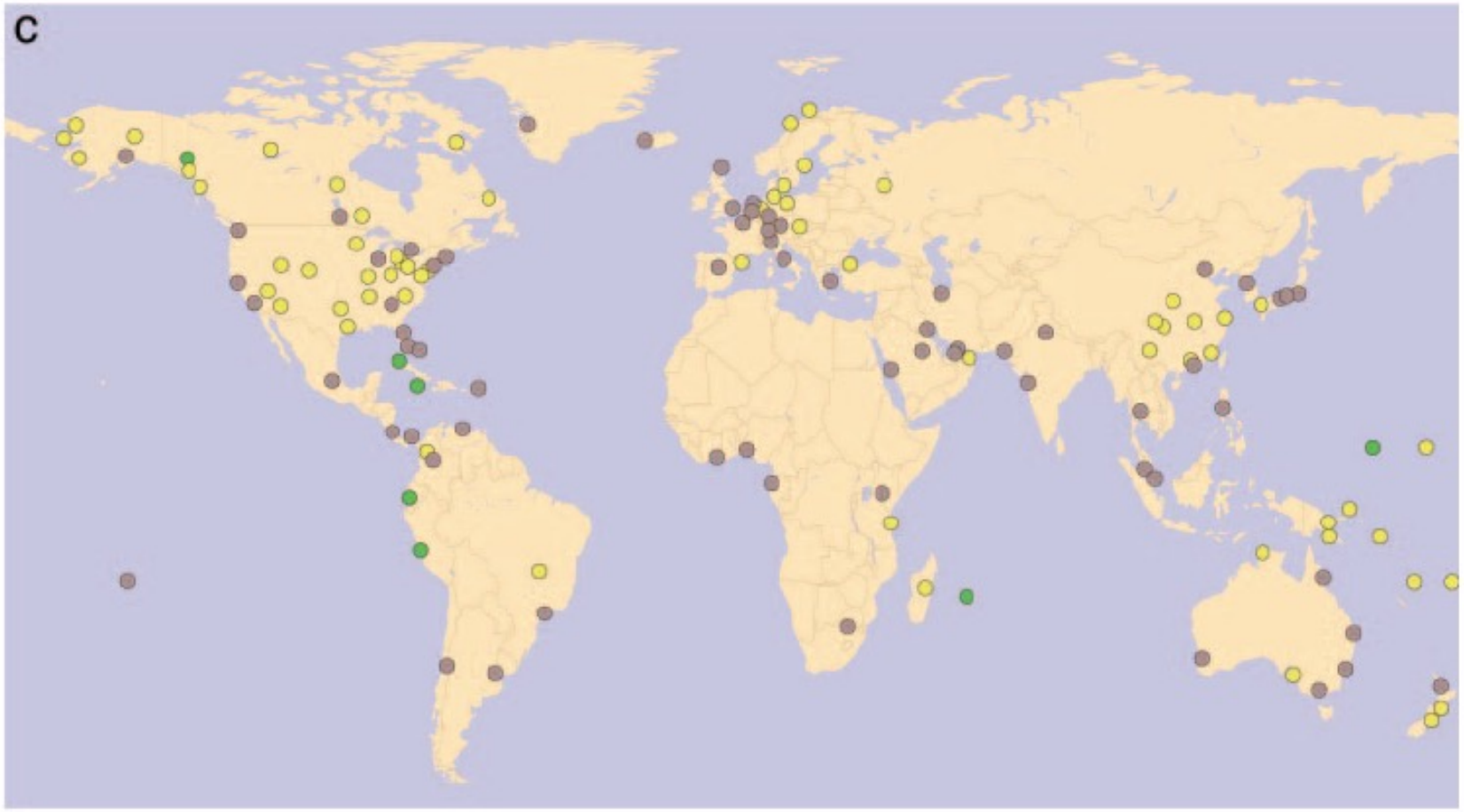
seed : *integer, random_state, or None (default)*

Indicator of random number generation state. See Randomness.

Returns:

G : *graph*

The graph after double edge swaps.



(Nonhub connectors (green), provincial hubs (yellow), and connector hubs (brown) in the worldwide air transportation network

- Network metrics used:

Betweeness, Communities, comparing to Randomized Network.

- Metrics proposed:

Participation Coefficient and Zscore within communities

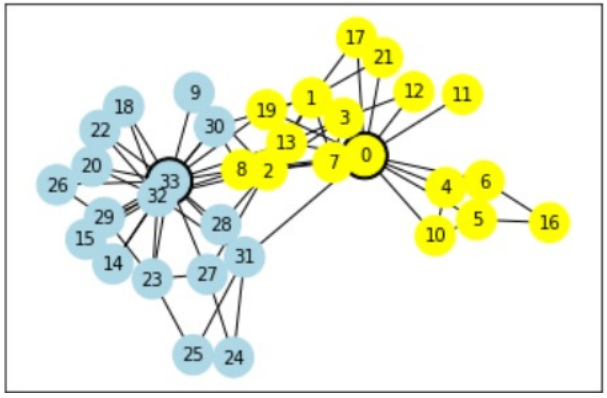
Network Story:
They are able to quantify non trivial role of airports
and to classify hundreds of airports in 6 groups.

They first noted that there were high
betweenness with low degree airports and found a way
to characterize the regional role of airports

Example network: Karate Club

8]:

```
karate = nx.karate_club_graph()
#Coloring factions
(faction1_col, faction2_col) = ('yellow','lightblue')
#Building list of colors
node_color = [''] * len(karate.nodes())
node_dict = dict(karate.nodes(data=True))
#Assign colors based on factions
for n in karate.nodes():
    if node_dict[n]['club'] == 'Mr. Hi':
        node_color[n] = faction1_col
    elif node_dict[n]['club'] == 'Officer':
        node_color[n] = faction2_col
    else:
        print("Something is broken")
        break;
# Initialize the position
pos = nx.spring_layout(karate,scale=0.2)
# Label nodes
nx.draw_networkx_labels(karate,pos,font_size=10,font_color='black')
# Highlight the leaders
nx.draw_networkx_nodes(karate,pos,{0,33},node_color=['black','black'],node_size=700)
# Now draw all the nodes, including leaders, using faction color scheme.
nx.draw_networkx_nodes(karate,pos,node_color=node_color,node_size=500)
nx.draw_networkx_edges(karate,pos)
plt.show()
```



Skip in the code the k—clique and Newman algorithm and go to:

Algorithm: Modularity Maximization Heuristics

(Louvain 2008) offers a percolation-inspired heuristic for community detection. The algorithm has two phases that are applied iteratively: the first phase is a greedy algorithm that places all nodes in their own community, then each node is moved to the community that gives the maximum gain in modularity. This is repeated for all nodes until no more modularity improvement can be made.

The second phase builds a new graph where the communities from phase 1 are collapsed into a single node, and the edges between communities collapse into single edges between the respective nodes with equivalent weight. Then phase 1 is repeated on this reduced graph, and the passes continue until no changes occur over a pass.

```
]: 1 ! pip install python-louvain
```

```
Requirement already satisfied: python-louvain in /Users/marta/opt/anaconda3/lib/python3.7/site-packages (0.15)
Requirement already satisfied: networkx in /Users/marta/opt/anaconda3/lib/python3.7/site-packages (from python-louvai
n) (2.6.2)
Requirement already satisfied: numpy in /Users/marta/opt/anaconda3/lib/python3.7/site-packages (from python-louvain)
(1.20.3)
```

```
]: 1
```

```
]: 1 import community as community_louvain
   2 import community
   3 from community import community_louvain
```

```
]: 1 partition = community_louvain.best_partition(karate)
   2 cc=[]
```

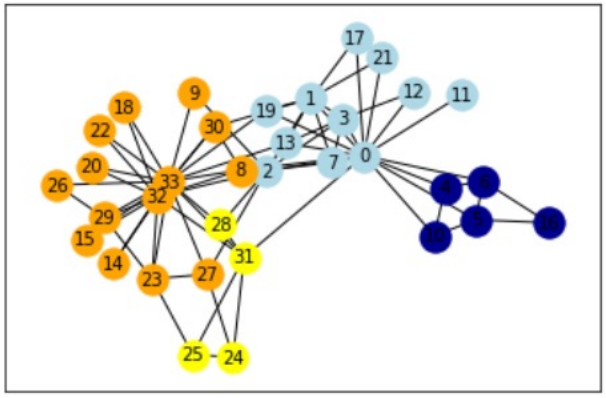
```
]: 1 partition
```

```
]: {0: 0,
    1: 0,
    2: 0,
    3: 0,
    4: 1,
    5: 1,
    6: 1,
    7: 0,
    8: 3,
    9: 3,
    10: 1,
    11: 0,
    12: 0,
    13: 0,
    14: 3,
    15: 3,
    16: 1,
```

Return a dictionary from nodes to Community Label

```
#drawing
size = float(len(set(partition.values())))
count = 0.
for com in set(partition.values()) :
    count = count + 1.
    list_nodes = [nodes for nodes in partition.keys()
                                if partition[nodes] == com]
    cc.append(list_nodes)


#Needs to know the number of partitions
#nx.draw_networkx(karate,pos)
#nx.draw_networkx_nodes(karate,pos,node_color=node_color)
nx.draw_networkx_nodes(karate,pos,nodelist=cc[0],node_color=['lightblue'])
nx.draw_networkx_nodes(karate,pos,nodelist=cc[2],node_color=['yellow'])
nx.draw_networkx_nodes(karate,pos,nodelist=cc[1],node_color=['darkblue'])
nx.draw_networkx_nodes(karate,pos,nodelist=cc[3],node_color=['orange'])
nx.draw_networkx_edges(karate,pos)
# Label nodes
nx.draw_networkx_labels(karate,pos,font_size=10,font_color='black')
plt.show()
```



Creates a list with the lists of nodes in each community

```
set(partition.values())
```

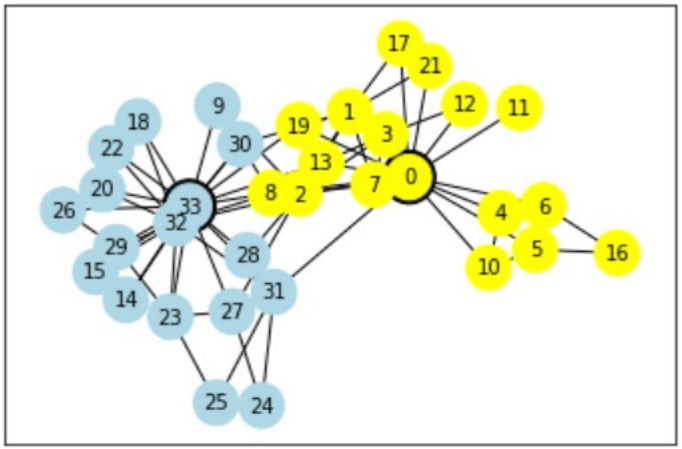
```
{0, 1, 2, 3}
```

```
cc
```

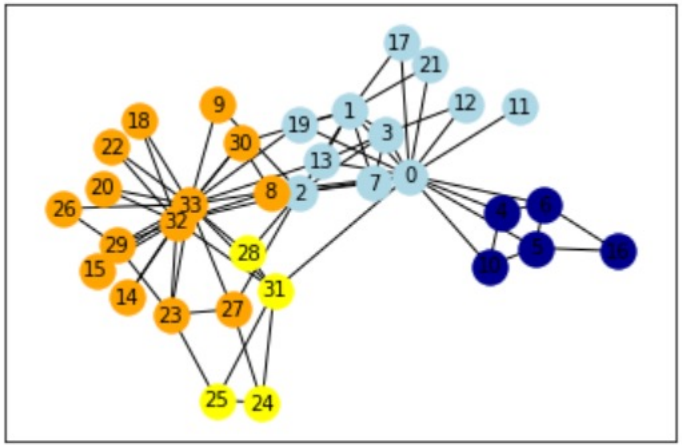
```
[[0, 1, 2, 3, 7, 11, 12, 13, 17, 19, 21],
 [4, 5, 6, 10, 16],
 [24, 25, 28, 31],
 [8, 9, 14, 15, 18, 20, 22, 23, 26, 27, 29, 30, 32, 33]]
```

You can extract subgraphs with this set of nodes only

Actual communities



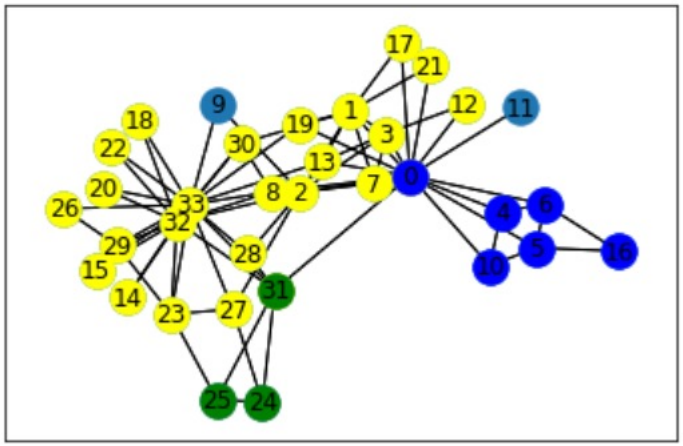
Louvain Labeling



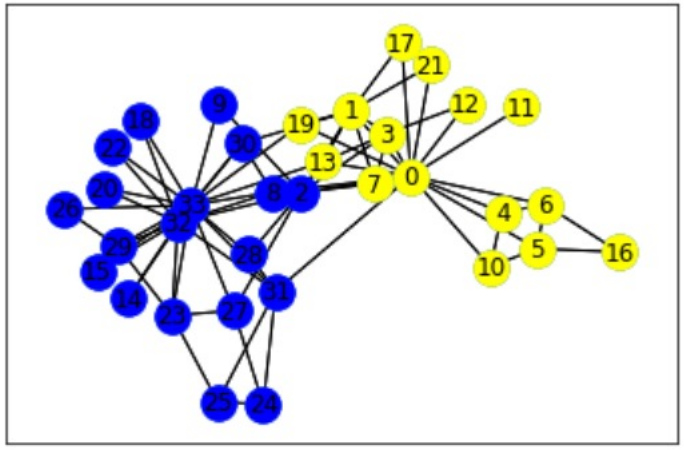

```
1  community_louvain.modularity(partition,karate)
```

```
0.41978961209730437
```

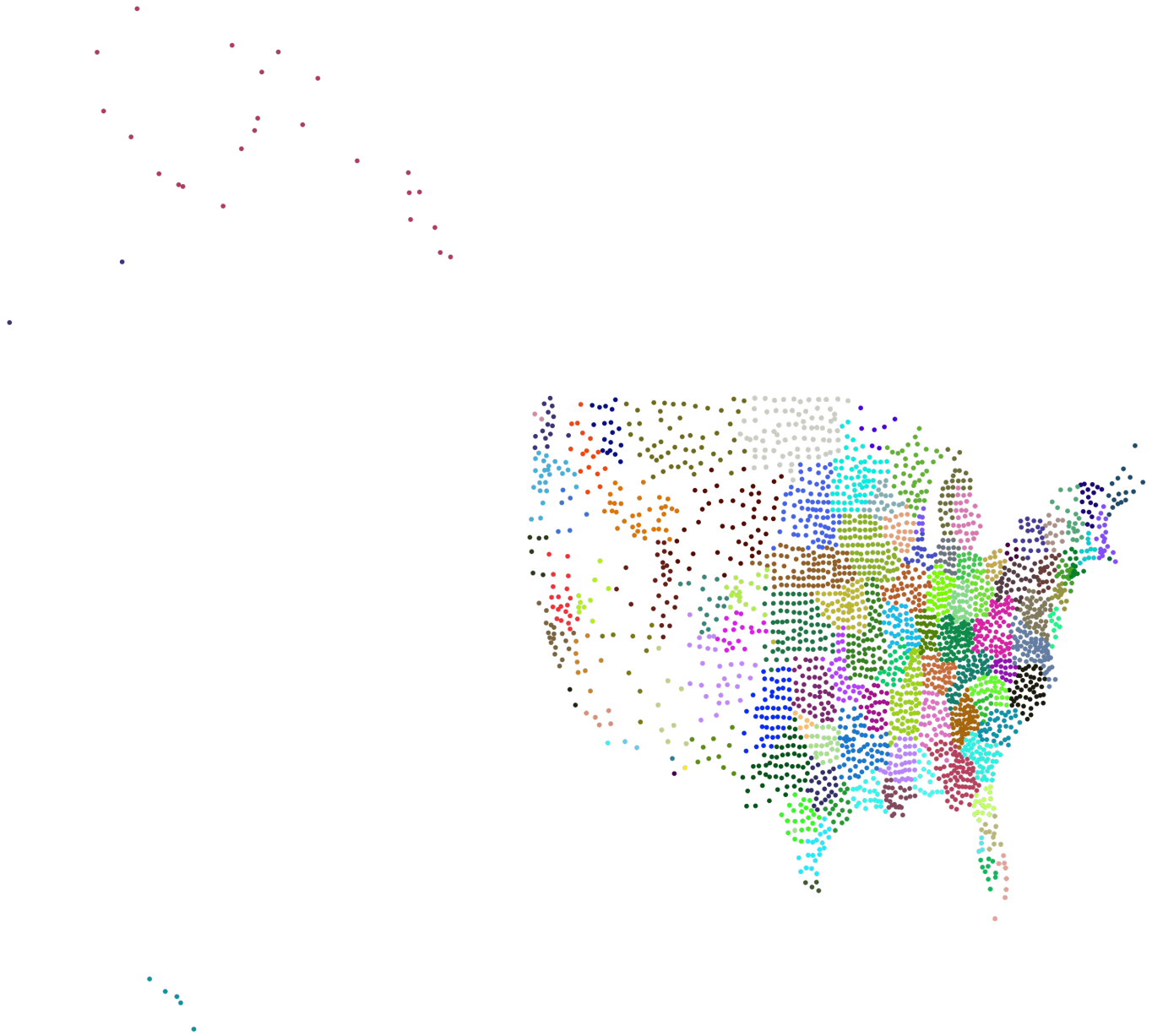
K-clique Method



Newman Method



Let's calculate communities from larger datasets



```
nodes = np.loadtxt("THE_NODES.txt",dtype=[('f0', '|S5'), ('f1', int),('f2', float), ('f3', float),('f4',int)],usec
links = np.loadtxt("THE_LINKS.txt",dtype=[('f0', '|S5'), ('f1', '|S5'),('f2', float), ('f3', int)],usecols=(0,1,2,
```

```
# build commuter flow network
g = nx.Graph()
for row in links:
    g.add_edge(row[0],row[1],weight=row[3])

# a) How many nodes and links are there?
print(nx.number_of_nodes(g))
print(nx.number_of_edges(g))
```

Add edges in a Graph
The Louvain algorithm
Only works in directed graph

```
3141
130909
```

```
poss = {}
for row in nodes:
    #print node
    node=row[0]
    lat=row[2]
    lon=row[3]
    poss[node] = (lon,lat)
```

Create a dictionary pos with Coordinates to display a spatial Network in the network draw function

Same code as before, create list of list of nodes

```
#first compute the best partition
partition = community_louvain.best_partition(g)
```

```
CC=[]
#drawing
size = float(len(set(partition.values())))
count = 0.
for com in set(partition.values()) :
    count = count + 1.
    list_nodes = [nodes for nodes in partition.keys()
                                    if partition[nodes] == com]
    CC.append(list_nodes)
```

```
count
```

```
108.0
```

```
community_louvain.modularity(partition,g)
```

```
0.9649983641163967
```

→ Great value of Modularity

```
plt.figure(figsize=(100,100))
for i in range(0,len(CC)):
    color=(random.random(),random.random(),random.random())
    nx.draw_networkx_nodes(g,poss,nodelist=CC[i],node_size=400,node_color=color)
plt.axis('off')
plt.savefig('CommunitiesUSACommutNet.png', bbox_inches='tight')
```

Saves figure

Network tools learned

- Data vs. Models comparisons
- Spatial Analysis Census Data
- K-means clustering
- Closeness, Betweenness, Degree
- K-Core extraction
- Robustness to Attacks and Failures
- Network Clustering

03/07	Lecture 8	Centralities and Network Resilience			
03/14	Lecture 9	Clustering Networks	Assignment 3_II		
03/21	Lecture 10	OSMNx + Guest Lecture M&G			
03/28	Spring Break	-	-		
04/04	Lecture 11	Discussion Project Ideas	Assignment 4 (Project Preparation)		
04/11	Lecture 12	Scaling in Transportation Networks			
04/18	Lecture 13	Dynamics on Networls	Assignment 5 (Paper Draft)		
04/25	Lecture 14	Class Summary	TBC		
05/02	Project Presentations				
05/03	Project Presentations				
05/04	Project Presentations				

Note: Present your project idea next class if you want to recruit Group members

Possible Data sets for projects:

https://docs.google.com/presentation/d/1abWEBqMEDpuvWF_TRVMKc0O6B6lekgt3EevSeYn7mp0/edit#slide=id.g2007d89a95e_0_33

For Next Class, start assignment 4 and participate for next class sharing in class about your project interests:

https://docs.google.com/presentation/d/1UVC7nIhjEO6kOWPSU8mVoKvc3Ye1SoPbGGQZRft1M68/edit#slide=id.p