MCS lab2

Urlich Icimpaye, Jakob Häggström, Oscar Jacobson

May 2022

1 Introduction

In this Lab we're going to examine and compare the robustness of a real world network and random networks. Thereafter we'll gather some statistics of the real world network and compare it to a random network which is going to be used as a null model. The real world network that was chosen was a data set of US airlines which has 332 vertices and 2126 edges. While the random network used is was a erdös-renyi model.

2 Robustness of networks

To test the robustness, measurements of the global clustering coefficient was taken while systematically removing a proportion, α, of edges from the network following some criteria. α ranged from 10 percent to 90 percent with increments of 10 percent. For the last part of this section α proportions of nodes was removed. To calculate the global clustering coefficients, the function *transitivity* was used which is given by the networkx package [1] and it's calculated as such:

$$T = 3\frac{\#triangles}{\#triads} \tag{1}$$

Where triads are possible triangles that have two edges with a shared vertex.

Firstly α proportion of edges were removed randomly by drawing from a uniform distribution. Where each edge had the same probability of being drawn. Thereafter edges was removed by choosing the vertex of max degree then choosing a random edge incident with that vertex. Last two criteria ,was that α proportion of nodes was removed by choosing the node with the highest degree and the opposite being the node with the lowest degree. Regarding the random network, simulations was repeated 10 times for every α

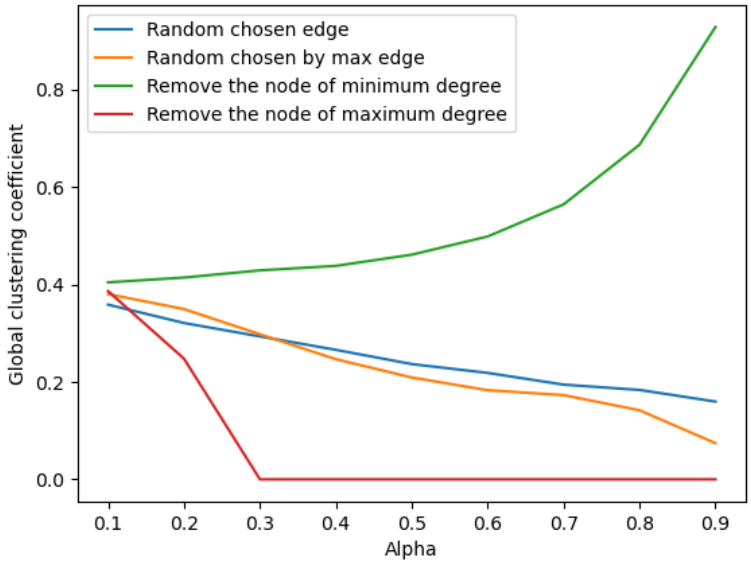

Figure 1: Global clustering coefficient vs α for real network

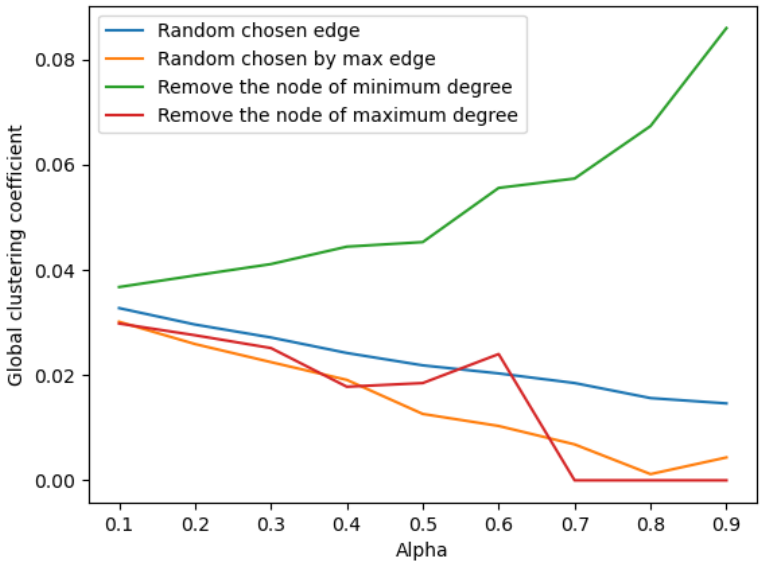

Figure 2: Global clustering coefficient vs α for random network, Erdos-Renyi in this case.

From figure 1 one sees that choosing random edges for removal had the same effect as choosing the random edge by the node with the maximum edges. But when removing nodes based of the node with the maximum edges, the global clustering coefficient goes towards zero. This is likely due to the fact that that the larger airlines in terms of edges are connected to other airlines. While removing the nodes with the least edges yielded a higher clustering coefficient. This is likely due to the fact that the number of triads are decreasing as seen in equation1. For

the random model the same behaviour can be seen from figure 2 as in 1. But the clustering coefficient goes to zero when α is 70 percent. This is might be because edges are more uniformly distributed between nodes. Although they have the same behavior as the real network when removing nodes with the lowest degree.

3 Random graphs as null models for networks

This section uses random graphs in order to examine the null hypothesis for the maximum modularity of the Airlines network. The maximum modularity is in this project calculated by using the networkx function *greedy_modularity_communities*, which uses Clauset-Newman-Moore algorthm in order to find the community with the maxium modularity, and the modularity of that particular community is found by using *modularity*[2]. The random graphs that are used in this project are Erdos-Renyi as well as configuration networks.

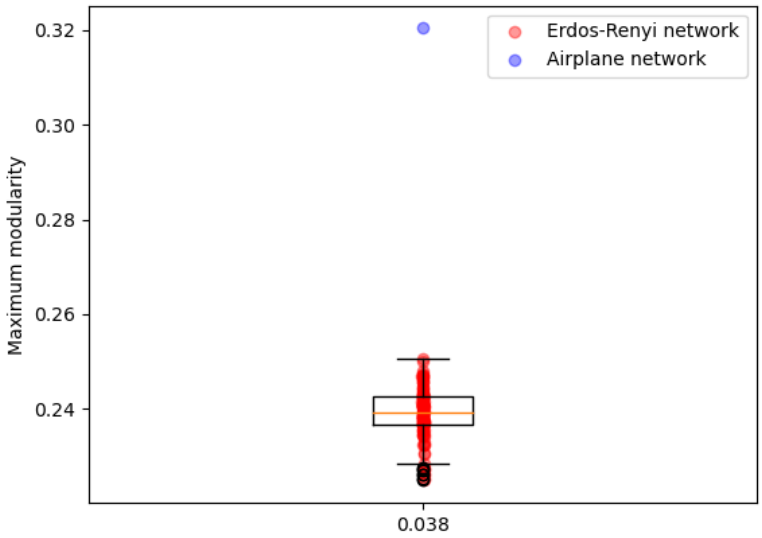

Figure 3: Box whisker plot with the maximum modularity of 100 Erdos-Renyi graphs versus the real network value.

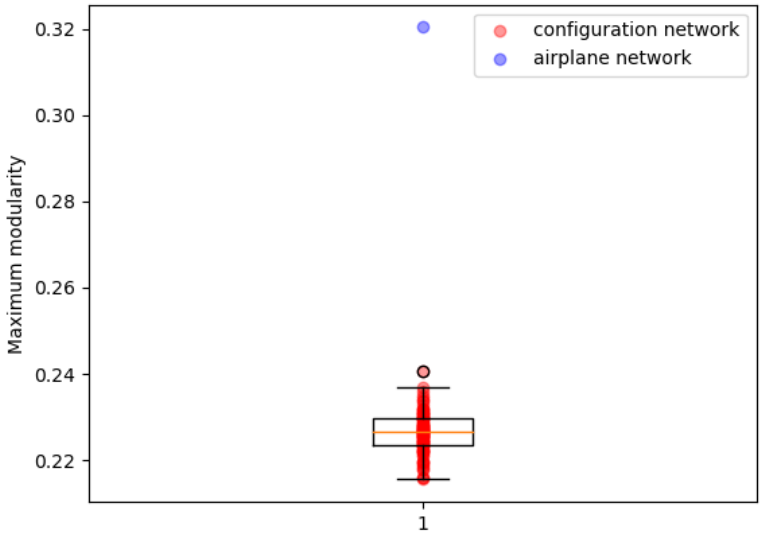

Figure 4: Box whisker plot with the maximum modularity of 100 Configuration graphs versus the real network value.

As seen in figure 3-4, the maximum modularity of the Airlines network are larger than 95% of the random samples for both random graphs, and we can discard the null-hypothesis in both cases. Though the scale of both figures are rather misleading, since the numerical value of the test statistic is close to the 5 percent largest observed values for both simulations.

References

[1] Transitivity. Available from: `https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.transitivity.html`

[2] *greedy_modularity_communities*. Available from: `https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html`

Appendix

```
#! /usr/bin/python3

import random

import networkx as nx
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
```

```
import os
from statistics import median_high

def get_subset(set, alpha):
    indicies = list(range(len(set)))
    return set[np.random.choice(indicies, size = round(alpha * len(set)))]

# ============== read in data from a file ==============

MAIN_DIR = os.path.split(os.path.abspath(__file__))[0]
data_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'data'))
fig_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'figs'))
print(MAIN_DIR)

df = open(os.path.join(data_dir, 'airlines.txt'), 'r')
num_vertices = 0

for line in df:
    line_list = line.strip().split()
    if line_list[0] == '*Vertices':
        num_vertices = int(line_list[1])
        break

print("num_vertices =", num_vertices)
G = nx.empty_graph(num_vertices)

reading_edges = False
for line in df:
    line_list = line.strip().split()
    if not reading_edges:
        if line_list[0] != "*Edges":
            continue
        else:
            reading_edges = True
            continue
    else:
        G.add_edge(int(line_list[0]), int(line_list[1]))

alpha_list = np.linspace(0.1,0.9,9)
n = G.order()
ne = G.size()
degree_dict = dict(G.degree())
max_deg = max(degree_dict, key = degree_dict.get)
glob_clust_list = []

for alpha in alpha_list:
    tempres = 0
    for i in range(10):
        edge_subset = get_subset(np.array(list(G.edges())),alpha)
        temp_G = G.copy()
        temp_G.remove_edges_from(edge_subset)
        tempres += nx.transitivity(temp_G)
    glob_clust_list.append(tempres / 10)


glob_clust_list2 = []
for alpha in alpha_list:
    tempres = 0
    for i in range(10):
        temp_G = G.copy()
```

```
69        stop_cond = round(len(temp_G.edges()) * alpha)
70
71        for n in range(stop_cond):
72           degree_dict = dict(temp_G.degree())
73           max_deg_key = max(degree_dict, key = degree_dict.get)
74           edges = list(temp_G.edges(nbunch = max_deg_key))
75           rand_ind = np.random.randint(0, len(edges)- 1)
76           temp_G.remove_edge(edges[rand_ind][0], edges[rand_ind][1])
77
78        tempres += nx.transitivity(temp_G)
79
80     glob_clust_list2.append(tempres / 10)
81
82 glob_clust_list3 = []
83 for alpha in alpha_list:
84
85     temp_G = G.copy()
86     stop_cond = round(len(temp_G.nodes()) * alpha)
87
88     for n in range(stop_cond):
89
90        degree_dict = dict(temp_G.degree())
91
92        max_deg_key = min(degree_dict, key = degree_dict.get)
93        print(max_deg_key)
94        temp_G.remove_node(max_deg_key)
95
96
97     glob_clust_list3.append(nx.transitivity(temp_G))
98
99
100
101
102 glob_clust_list4 = []
103 for alpha in alpha_list:
104
105     temp_G = G.copy()
106     stop_cond = round(len(temp_G.nodes()) * alpha)
107
108     for n in range(stop_cond):
109
110        degree_dict = dict(temp_G.degree())
111
112        max_deg_key = max(degree_dict, key = degree_dict.get)
113        print(max_deg_key)
114        temp_G.remove_node(max_deg_key)
115
116
117     glob_clust_list4.append(nx.transitivity(temp_G))
118
119
120
121
122
123
124 plt.figure()
125 plt.plot(alpha_list, glob_clust_list, label = 'Random␣chosen␣edge')
126 plt.plot(alpha_list, glob_clust_list2,label = 'Random␣chosen␣by␣max␣edge')
127 plt.plot(alpha_list, glob_clust_list3,label = 'Remove␣the␣node␣of␣minimum␣degree')
128 plt.plot(alpha_list, glob_clust_list4 ,label = 'Remove␣the␣node␣of␣maximum␣degree')
```

```
plt.ylabel('Global clustering coefficient')
plt.xlabel('Alpha')
plt.legend()
plt.savefig(os.path.join(fig_dir, 'ass1_real.png'))
plt.show()

print("G is of order", n, "and size", ne, "and maximum degree is", max_deg) # / just for info

# Plot the network:
#nx.draw(G, with_labels=False, node_color='orange', node_size=30, / edge_color='black', linewidths=1, font_size=15)

# ============== Community Partition of airline network =============

# from networkx.algorithms import community

# G_communities=community.greedy_modularity_communities(G)
# mod=community.modularity(G,G_communities)
# print("Modularity value of this partition on airline network:",mod)


# G_comm=sorted(map(sorted, G_communities))


# #Plot the graph with node colours showing community membershiop
# node_colors_map = {}
# for i, lg in enumerate(G_comm):
# for node in lg:
# node_colors_map[node] = i
# node_colors = [node_colors_map[n] for n in G.nodes]


# #fixes a layout of the nodes (note re-running this will change the layout)
# pos = nx.fruchterman_reingold_layout(G)
# print('matplotlib: {}'.format(matplotlib.__version__))

# nx.draw(G, pos=pos, with_labels=False, node_color=node_colors, linewidths=1, / font_size=15,node_size=30)
# plt.show()
```

Listing 1: The code that generated figure 1

```
#! /usr/bin/python3

import random

import networkx as nx
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import os
from statistics import median_high

MAIN_DIR = os.path.split(os.path.abspath(__file__))[0]
data_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'data'))
fig_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'figs'))

def get_subset(set, alpha):
   indicies = list(range(len(set)))
   return set[np.random.choice(indicies, size = round(alpha * len(set)))]
```

```
19
20 # ============== read in data from a file ==============
21
22
23
24 df = open(os.path.join(data_dir, 'airlines.txt'), 'r')
25 num_vertices = 0
26
27 for line in df:
28    line_list = line.strip().split()
29    if line_list[0] == '*Vertices':
30       num_vertices = int(line_list[1])
31       break
32
33 print("num_vertices = ", num_vertices)
34 G = nx.empty_graph(num_vertices)
35
36 reading_edges = False
37 for line in df:
38    line_list = line.strip().split()
39    if not reading_edges:
40       if line_list[0] != "*Edges":
41          continue
42       else:
43          reading_edges = True
44          continue
45    else:
46       G.add_edge(int(line_list[0]), int(line_list[1]))
47
48
49 alpha_list = np.linspace(0.1,0.9,9)
50 n = G.order()
51 ne = G.size()
52 degree_dict = dict(G.degree())
53
54 max_deg = max(degree_dict, key = degree_dict.get)
55 rand_G = nx.erdos_renyi_graph(n, ne / ((n * (n - 1) / 2)))
56 glob_clust_list = []
57
58 for alpha in alpha_list:
59    tempres = 0
60    for i in range(10):
61       edge_subset = get_subset(np.array(list(rand_G.edges())),alpha)
62       temp_G = rand_G.copy()
63       temp_G.remove_edges_from(edge_subset)
64       tempres += nx.transitivity(temp_G)
65    glob_clust_list.append(tempres / 10)
66
67
68 glob_clust_list2 = []
69 for alpha in alpha_list:
70    tempres = 0
71    for i in range(10):
72       temp_G = rand_G.copy()
73       stop_cond = round(len(temp_G.edges()) * alpha)
74
75       for n in range(stop_cond):
76          degree_dict = dict(temp_G.degree())
77          max_deg_key = max(degree_dict, key = degree_dict.get)
78          edges = list(temp_G.edges(nbunch = max_deg_key))
```

```
            rand_ind = np.random.randint(0, len(edges)- 1)
            temp_G.remove_edge(edges[rand_ind][0], edges[rand_ind][1])

        tempres += nx.transitivity(temp_G)

    glob_clust_list2.append(tempres / 10)

glob_clust_list3 = []
for alpha in alpha_list:

    temp_G = rand_G.copy()
    stop_cond = round(len(temp_G.nodes()) * alpha)

    for n in range(stop_cond):

        degree_dict = dict(temp_G.degree())

        max_deg_key = min(degree_dict, key = degree_dict.get)
        print(max_deg_key)
        temp_G.remove_node(max_deg_key)


    glob_clust_list3.append(nx.transitivity(temp_G))




glob_clust_list4 = []
for alpha in alpha_list:

    temp_G = rand_G.copy()
    stop_cond = round(len(temp_G.nodes()) * alpha)

    for n in range(stop_cond):

        degree_dict = dict(temp_G.degree())

        max_deg_key = max(degree_dict, key = degree_dict.get)
        print(max_deg_key)
        temp_G.remove_node(max_deg_key)


    glob_clust_list4.append(nx.transitivity(temp_G))







plt.figure()
plt.plot(alpha_list, glob_clust_list, label = 'Random chosen edge')
plt.plot(alpha_list, glob_clust_list2,label = 'Random chosen by max edge')
plt.plot(alpha_list, glob_clust_list3,label = 'Remove the node of minimum degree')
plt.plot(alpha_list, glob_clust_list4 ,label = 'Remove the node of maximum degree')
plt.ylabel('Global clustering coefficient')
plt.xlabel('Alpha')
plt.legend()
plt.savefig(os.path.join(fig_dir, 'ass1_rand.png'))
plt.show()

```

```
print("G is of order", n, "and size", ne, "and maximum degree is", max_deg) # / just for info

# Plot the network:
#nx.draw(G, with_labels=False, node_color='orange', node_size=30, / edge_color='black', linewidths=1, font_size=15)

# ============== Community Partition of airline network =============

# from networkx.algorithms import community

# G_communities=community.greedy_modularity_communities(G)
# mod=community.modularity(G,G_communities)
# print("Modularity value of this partition on airline network:",mod)


# G_comm=sorted(map(sorted, G_communities))


# #Plot the graph with node colours showing community membershiop
# node_colors_map = {}
# for i, lg in enumerate(G_comm):
# for node in lg:
# node_colors_map[node] = i
# node_colors = [node_colors_map[n] for n in G.nodes]


# #fixes a layout of the nodes (note re-running this will change the layout)
# pos = nx.fruchterman_reingold_layout(G)
# print('matplotlib: {}'.format(matplotlib.__version__))

# nx.draw(G, pos=pos, with_labels=False, node_color=node_colors, linewidths=1, / font_size=15,node_size=30)
# plt.show()
```

Listing 2: The code that generated figure 2

```
import random
from networkx.algorithms import community
import networkx as nx
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import os
from statistics import median_high

def get_subset(set, alpha):
    indicies = list(range(len(set)))
    return set[np.random.choice(indicies, size = round(alpha * len(set)))]

# ============== read in data from a file ============== These lines are taken / from the labs.

MAIN_DIR = os.path.split(os.path.abspath(__file__))[0]
data_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'data'))
fig_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'figs'))
print(MAIN_DIR)

df = open(os.path.join(data_dir, 'airlines.txt'), 'r')
num_vertices = 0

```

```
for line in df:
   line_list = line.strip().split()
   if line_list[0] == '*Vertices':
      num_vertices = int(line_list[1])
      break

print("num_vertices =", num_vertices)
G = nx.empty_graph(num_vertices)

reading_edges = False
for line in df:
   line_list = line.strip().split()
   if not reading_edges:
      if line_list[0] != "*Edges":
         continue
      else:
         reading_edges = True
         continue
   else:
      G.add_edge(int(line_list[0]), int(line_list[1]))

# ============== read in data from a file ==============

n = G.order()
ne = G.size()
G_communities=community.greedy_modularity_communities(G)
mod_star=community.modularity(G,G_communities)

print(f"real value: {mod_star}")

t_n = []
p_n = []
prob = ne / ((n * (n - 1))/2)

for i in range(100):
   rand_G = nx.erdos_renyi_graph(n = n, p = prob)
   G_communities_temp=community.greedy_modularity_communities(rand_G)
   mod_com_temp=community.modularity(rand_G,G_communities_temp)
   t_n.append(mod_com_temp)
   p_n.append(rand_G.size()/((n * (n - 1))/2))


p = round(prob,3)
plt.figure()
plt.boxplot(t_n, positions= [p])
plt.scatter(p_n, t_n, c = 'r', alpha = 0.4, label = 'Erdos-Renyi network')
plt.scatter(p , mod_star, c = 'b', alpha = 0.4, label = 'Airplane network')
plt.ylabel('Maximum modularity')
plt.legend()
plt.savefig(os.path.join(fig_dir, 'erdos_graph.png'))
plt.show()
```

Listing 3: The code that generated figure 3

```
import random
from networkx.algorithms import community
import networkx as nx
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import os
```

```
from statistics import median_high

def get_subset(set, alpha):
    indicies = list(range(len(set)))
    return set[np.random.choice(indicies, size = round(alpha * len(set)))]

# ============== read in data from a file ==============

MAIN_DIR = os.path.split(os.path.abspath(__file__))[0]
data_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'data'))
fig_dir = os.path.abspath(os.path.join(MAIN_DIR,'..', 'figs'))
print(MAIN_DIR)

df = open(os.path.join(data_dir, 'airlines.txt'), 'r')
num_vertices = 0

for line in df:
    line_list = line.strip().split()
    if line_list[0] == '*Vertices':
        num_vertices = int(line_list[1])
        break

print("num_vertices =", num_vertices)
G = nx.empty_graph(num_vertices)

reading_edges = False
for line in df:
    line_list = line.strip().split()
    if not reading_edges:
        if line_list[0] != "*Edges":
            continue
        else:
            reading_edges = True
            continue
    else:
        G.add_edge(int(line_list[0]), int(line_list[1]))

n = G.order()
ne = G.size()

G_communities=community.greedy_modularity_communities(G)
mod_star=community.modularity(G,G_communities)

print(f"{mod_star}")
degree_sequence = [d for n, d in G.degree()]
t_n = []
p_n = []

for i in range(100):
    rand_G = nx.configuration_model(degree_sequence)
    rand_G = nx.Graph(rand_G)

    G_communities_temp=community.greedy_modularity_communities(rand_G)
    mod_com_temp=community.modularity(rand_G,G_communities_temp)

    t_n.append(mod_com_temp)


plt.figure()
plt.boxplot(t_n)
```

```
68 plt.scatter(np.ones(100) , t_n, c = 'r', alpha = 0.4, label = 'configuration␣/
       network')
69 plt.scatter([1] , mod_star, c = 'b', alpha = 0.4, label = 'airplane␣network')
70 plt.ylabel('Maximum␣modularity')
71 plt.legend()
72 plt.savefig(os.path.join(fig_dir, 'config_graph.png'))
73 plt.show()
```

Listing 4: The code that generated figure 4