Report on Analysis of Networks of American College Football Games and Coauthorships in High-Energy Physics Theory

Haowei CHEN

Hong Kong University of Science and Technology

Abstract. Real-world networks are useful models for data analysis. In this report, we select two certain networks: the network of American College Football Games and the network of High-Energy Physics Theory Coauthorships as representatives. We calculated their basic parameters, detected communities in them, and performed attacks to test their network resilience. In the final section of the report, we analyze our results and determine the special network classes that they are in alternately by the properties of these two networks.

Keywords: Complex Networks · Community Detection · Network Resilience.

1 Introduction

The network of American College Football Games is a small network with only 115 nodes and 613 edges, which denotes the American football games between Division IA colleges during the regular season in Fall 2000. For convenience, it is hereinafter referred to as Network 1. This dataset is provided in [1].

The network of coauthorships between scientists in High-Energy Physics is a large weighted network that shows coauthorships between scientists in High-Energy Physics posting preprints on the High-Energy Theory E-Print Archive between Jan 1, 1995 and December 31, 1999. Since it is disconnected, we choose the largest connected component of it to analyse. This component has 5835 nodes and 13815 edges. For convenience, it is hereinafter referred to as Network 2. This dataset is provided in [2].

In this report, we discuss some topological and statistical properties of these two networks above. The rest of this report is organized as follows. In Section 2, we detect some basic parameters of these networks. Then we detect the communities in two networks in Section 3 and test the network resilience of the large network in Section 4. Finally, in Section 5, we discuss about the special classes that these networks belong to alternatively.

2 Basic Parameters

In this section, we calculate three basic parameters of networks: degree distribution, the average shortest distance, and the average clustering coefficient.

The following figure shows the degree distribution of Network 1:

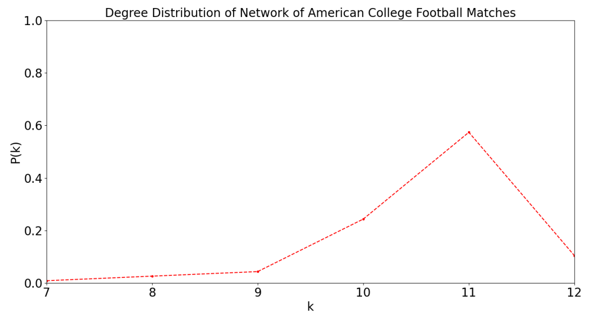

Fig. 1. Figure for Degree Distribution of American Football Games

The average shortest distance of this network is 2.51, and the average clustering coefficient of this network is 0.40.

For Network 2, in order to see the distribution clearly, we show its degree distribution by a lin-log graph as follows:

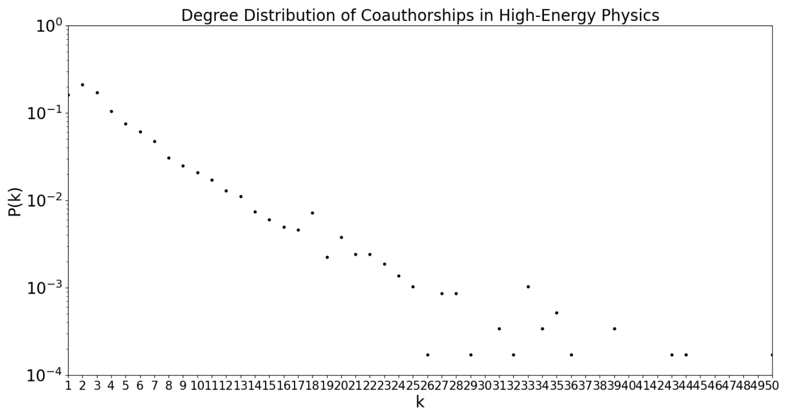

Fig. 2. Lin-log Plot for Degree Distribution of Coauthorships in High-Energy Physics

The average shortest distance of this network is 7.03, and the average clustering coefficient of this network is 0.51.

3 Community Detection

In this section, we detect the communities in two networks. In order to obtain the best visual effect, we apply different algorithms to these two networks.

3.1 American College Football Games

For Network 1, we use the Girvan-Newman algorithm [1] to find its communities. We calculate the modularities of all partitions into different number of communities we got via the Girvan-Newman algorithm first. The result obtained is visualized by the figure below:

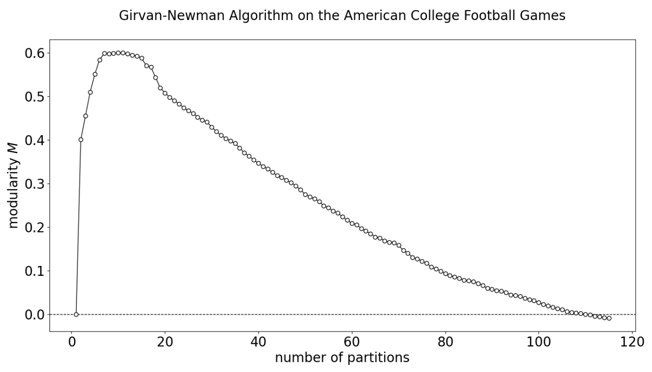

Fig. 3. Modularities of Partitions Obtained by Girvan-Newman Algorithm on American College Football Games

As is shown in the figure above, the modularity touches its maximum when the number of communities is 9 or 10. Hence, we partition the network into 10 communities. The result is visualized as shown below:

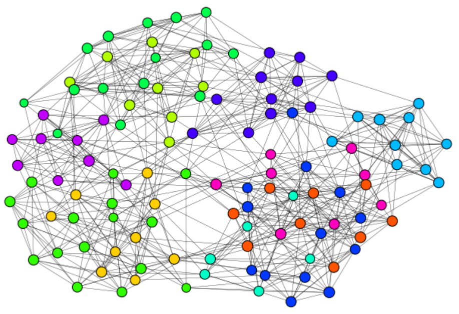

Fig. 4. Girvan-Newman Algorithm on American College Football Games

3.2 Coauthorships in High-Energy Physics Theory

For Network 2, we use the Clauset-Newman-Moore algorithm [3] to detect its communities. We obtain 98 communities in total. This result is visualized with the network below:

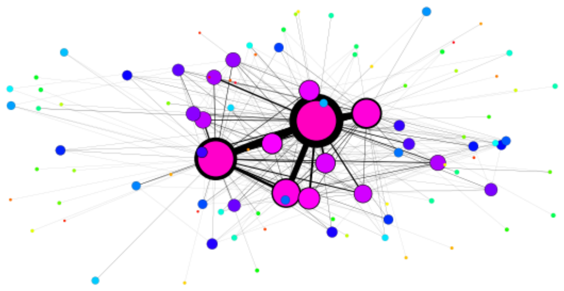

Fig. 5. Community Graph of Network for Coauthorships in High-Energy Physics

4 Network Resilience

By performing a random attack, we obtain the following figure of the size of the giant component against the fraction of the removed nodes for Network 2 as follows:

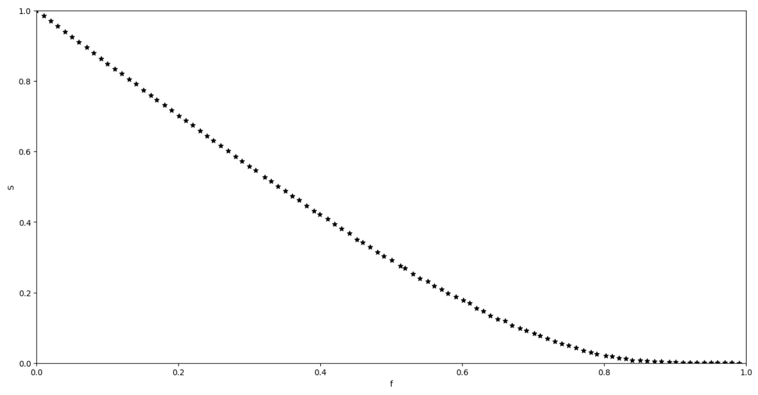

Fig. 6. Size of the Giant Component Against the Fraction of Nodes Removed

For targeted attacks, we choose the strategy to attack the vertices with the largest degrees. By keeping enlarging the propotion of nodes removed, we obtain the figure as below:

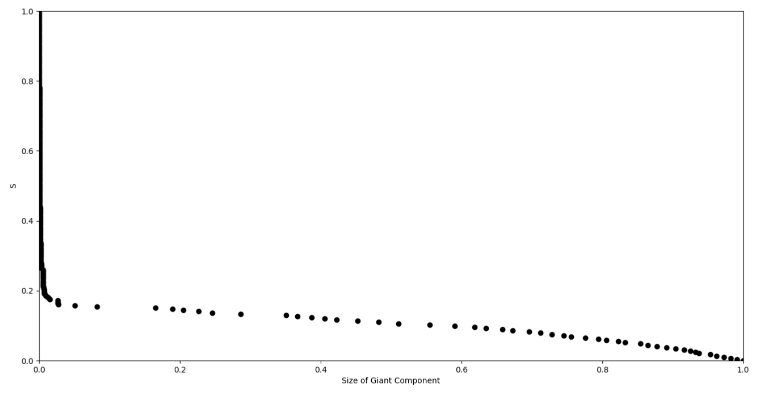

Fig. 7. Size of the Giant Component Against the Fraction of Nodes Removed

The result shows that this network can resist random attacks well but is sensitive to targeted attacks.

5 Discussions

In this section, we analyze the properties of two networks and discuss the classes of networks they are in.

5.1 American College Football Games

This small network appears to be very similar to a regular network. The degree distribution of it is concentrated and the standard deviation of the degrees is only 5. Moreover, its average shortest distance is very small. This implies that though for each node in this network, most nodes are not its neighbors but they can be reached by a small number of steps. Moreover, it has a clustering coefficient that is somehow high. Hence, this network is a small-world network.

From the community detection results, we can observe that the sizes of communities are very close. In fact, the size of the largest community in this network is 16, while the smallest community in this network has 6 nodes. In fact, by printing out all communities of this network, we can find that the communities detected are very similar with different conferences of American football teams. The following table shows the comparison of part of community results and conferences of American football teams. [5]

Table 1. Comparison Table for Communities and Conferences

Community	**Conference**	**Same Teams Proportion(Community)**
1	MountainWest	87.5%
2	ACC	88.89%
3	Big Ten	100%
4	Big 12	61.54%
5	Pac-12	100%
6	Mid-America	66.67%

Since this network was constructed based on matches in 2000, in the past over 20 years there were many changes of conferences and American college football teams, this already provides convincing evidence of the fact that communities match conferences of American college football teams. According to the rules of American college football games, [5] teams in the same conference are arranged more games against each other than teams from a different conference. This explains why teams in the same conference appear to link with each other closer in the network and be found in the same community.

5.2 Coauthorships in High-Energy Physics Theory

This network appears to be a scale-free network since in the network resilience tests, it turns out to be sensitive to targeted attacks but shows a high resilience towards random attacks. Moreover, its degree distribution shows that only very few nodes in it have a high degree while most of the nodes only have degrees less than 10. For a clear observation, we plot it in a loglog plot as below:

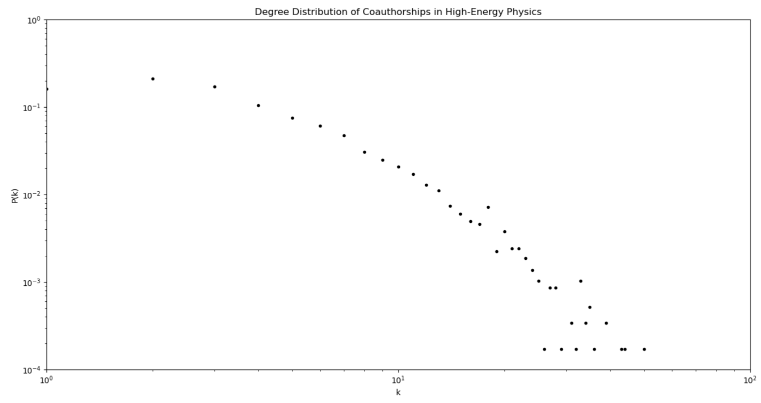

Fig. 8. Degree Distribution of Coauthorships in High-Energy Physics Theory In a Loglog Plot

As is shown above, the degree distribution figure is also similar with the degree distribution figure of a scale-free network.

From the community graph above, we can observe that in this network there are two extremely large communities. Moreover, these two communities are linked to each other very closely. This shows that authors in these communities are core contributors in this field. The output of the node with the highest degree in the largest community verifies this conclusion. That node is B. Zwiebach, a world-leading expert in string-field theory from MIT.

Acknowledgements

This report is written in the LaTeX template provided by LNCS homepage [4]. All figures and source codes used in this report can be found in the attached files of this report.

References

1. M. Girvan and M. E. J. Newman: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99**, 7821-7826 (2002). https://doi.org/10.1073/pnas.122653799
2. M. E. J. Newman: The structure of scientific collaboration networks. Proc. Natl. Acad. Sci. USA **98**, 404-409 (2001). https://doi.org/10.1073/pnas.98.2.404
3. Clauset A., Newman M. E. J., and Moore C. (2004) Finding community structure in very large networks, Phys. Rev. E **70**, 066111. https://doi.org/10.1103/PhysRevE.70.066111
4. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017
5. ESPN NCAAF Page, https://www.espn.com/college-football/teams

Appendix

6 Source Codes

6.1 Network for American College Football Games

The codes I use for the network for American College Football Games is as below:

```python
# -*- coding: utf-8 -*-
"""
MSDM5056 Proj1 Small Network

@author: 17100
"""


import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpt
from matplotlib.cm import gist_rainbow_r as cmap

data1 = nx.read_gml(r"C:\Users\17100\.spyder-py3\football.gml
    ")
n = data1.number_of_nodes()
m = data1.number_of_edges()
fig1 = plt.figure(figsize=(16, 8), dpi=100)
plt.title("Degree Distribution of Network of American College
     Football Matches", fontsize=20)
degrees = [k for m, k in data1.degree]
kmin, kmax = min(degrees), max(degrees)
kRng = np.arange(kmin, kmax+1)
dh = [m/n for m in nx.degree_histogram(data1)[kmin:kmax+1]]
plt.plot(kRng, dh, "r.--", mfc=None)
plt.xticks(np.linspace(kmin, kmax, kmax-kmin+1, True),
    fontsize=30)
plt.yticks(np.linspace(0, 1, 6, True), fontsize=20)
plt.xlabel("k", fontsize=20)
plt.ylabel("P(k)", fontsize=20)
plt.xlim(kmin, kmax)
plt.ylim(0, 1)
print("The number of nodes in this network is:{}".format(n))
print("The number of edges in this network is:{}".format(m))

print("The average shortest distance of this network is:{}".
    format(nx.average_shortest_path_length(data1)))
```

```
print("The average clustering coefficient of this network is
    :{}".format(nx.average_clustering(data1)))

data1.graph['GN'] = list(nx.community.girvan_newman(data1))
    [::-1]
fig2 = plt.figure(figsize=(16, 8), dpi=100)


def modularity(G, partitions):
    Qs = partitions
    E = G.number_of_edges()
    B = nx.modularity_matrix(G)
    C = np.array([[1 if i in Q else 0 for Q in Qs] for i in G
    ])
    return np.trace(C.T@B@C) / (2*E)

plt.axhline(0, c='k', ls='--', lw=1)

M = {len(Qs): modularity(data1, Qs) for Qs in data1.graph['GN
    ']}
M[1] = modularity(data1, (set(data1),))
plt.plot(M.keys(), M.values(), 'ko-', mfc='w', lw=1)

plt.xlabel('number of partitions', fontsize=20)
plt.ylabel('modularity $M$', fontsize=20)
plt.title('Girvan-Newman Algorithm on the American College
    Football Games\n', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()

data1.graph['pos'] = nx.kamada_kawai_layout(data1)
data1.graph['degs'] = nx.degree_centrality(data1)
for c, community in enumerate(data1.graph['GN'][-10]):
    for n in community: data1.nodes[n]['GN'] = c
cNum = c+1


def data1_plot(title, node_color):
    plt.figure(figsize=(9, 6))
    plt.title(title)
    nx.draw(data1, pos=data1.graph['pos'],
            width=.25, edgecolors='k', linewidths=.75,
            node_size=[k*1000 for k in data1.graph['degs'].
    values()],
            node_color=node_color)

data1_plot('the Girvan-Newman algorithm on the American
    College Football Games Network',
                [cmap(c/cNum) for n, c in data1.nodes('GN')])
```

6.2 Network for High-Energy Physics Theory Coauthorships

The codes I use for the network for the High-Energy Physics Theory Coauthorships is shown below:

```python
# -*- coding: utf-8 -*-
"""
MSDM5056 Proj 1

@author: 17100
"""


import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpt
from matplotlib.cm import gist_rainbow_r as cmap
import random as r

H = nx.read_gml("C:/Users/17100/.spyder-py3/hep-th.gml")
data1 = H.subgraph(max(nx.connected_components(H), key=len))
n = data1.number_of_nodes()
m = data1.number_of_edges()
fig1 = plt.figure(figsize=(16, 8), dpi=100)
plt.title("Degree Distribution of Coauthorships in High-
    Energy Physics", fontsize=20)
degrees = [k for m, k in data1.degree]
kmin, kmax = min(degrees), max(degrees)
kRng = np.arange(kmin, kmax+1)
dh = [m/n for m in nx.degree_histogram(data1)[kmin:kmax+1]]
plt.semilogy(kRng, dh, "k.", mfc=None)
plt.xticks(np.linspace(kmin, kmax, kmax-kmin+1, True),
    fontsize=20)
plt.yticks(fontsize=20)
plt.xlabel("k", fontsize=20)
plt.ylabel("P(k)", fontsize=20)
plt.xlim(kmin, kmax)
plt.ylim(10**(-4),1)
print("The number of nodes in this network is:{}".format(n))
print("The number of edges in this network is:{}".format(m))

print("The average shortest distance of this network is:{}".
    format(nx.average_shortest_path_length(data1)))
print("The average clustering coefficient of this network is
    :{}".format(nx.average_clustering(data1)))

fig2 = plt.figure(figsize=(16, 8), dpi=100)
data1.graph['pos'] = nx.kamada_kawai_layout(data1)
```

```python
data1.graph['degs'] = nx.degree_centrality(data1)
plt.title("Community Graph for Network of Coauthorships in High-Energy Physics", fontsize=20)
data1.graph['CNM'] = nx.community.greedy_modularity_communities(data1)
for c, community in enumerate(data1.graph['CNM']):
    for n in community: data1.nodes[n]['CNM'] = c
cNum = c+1
print("The number of communities in Network $2$ is:{}".format(cNum))


def community_network(G, algorithm):

    def edges_between(G, U, V):
        return sum(sum(G.has_edge(u, v) for u in U) for v in V)

    G_ = nx.Graph()
    for i, Q in enumerate(G.graph[algorithm]):
        intra = nx.subgraph(G, Q).number_of_edges()
        G_.add_node(i, size=len(Q), loop=intra)
    for i in G_:
        for j in range(i):
            inter = edges_between(
                G,G.graph[algorithm][i], G.graph[algorithm][j])
            if inter > 0: G_.add_edge(i, j, weight=inter)
    G_.graph['pos'] = {i: np.mean(
        [G.graph['pos'][n] for n in G.graph[algorithm][i]], axis=0) for i in G_}
    return G_


def community_diagram(G_, title, node_color):
    sizes = np.array([s for n, s in G_.nodes('size')])
    loops = np.array([l for n, l in G_.nodes('loop')])
    weights = np.array([w for u, v, w in G_.edges(data='weight')])

    plt.figure(figsize=(8, 4))
    plt.title(title)
    nx.draw(G_, pos=G_.graph['pos'], edgecolors='k',
            node_color=node_color,
            node_size=sizes/sizes.max()*1000,
            linewidths=loops/loops.max()*5,
            width=weights/weights.max()*5)

data1_CNM = community_network(data1, 'CNM')
community_diagram(data1_CNM,
                  'the CNM algorithm on the collaboration network',
                  [cmap(c/cNum) for c in data1_CNM])

vset = data1.nodes()


def rand_attack(H, p):
    for v in vset:
        c = [1]*int(1000*p) + [0]*int(1000*(1-p))
        d = r.choice(c)
        if d == 1:
            H.remove_node(v)
        else:
            continue

pset = np.linspace(0, 1, 100, False)
results = []
results2 = []
for p in pset:
    ssize = 0
    fremove = 0
    for i in range(100):
        W = data1.copy()
        rand_attack(W, p)
        ssize += W.subgraph(max(nx.connected_components(W), key=len, default=[])).number_of_nodes()
        fremove += 1-float(W.number_of_nodes())/data1.number_of_nodes()
    results.append((float(ssize)/n)/100)
    results2.append(float(fremove)/100)
fig3 = plt.figure(figsize=(16, 8), dpi=100)
plt.scatter(results2, results, color = "black", marker = "*")
plt.xlabel("f", fontsize=20)
plt.ylabel("S", fontsize=20)
plt.xlim(0, 1)
plt.ylim(0, 1)

h = nx.degree(data1)
T=[(d, m) for m, d in h]
degree_sequence = sorted(T, reverse=True)
T0=[m for d, m in degree_sequence]


def target_attack(W, k):
        W.remove_nodes_from(T0[:k])



results3 = []
```

```python
results4 = []
for i in range(0, len(degree_sequence), 20):
    ssize = 0
    fremove = 0
    for j in range(100):
        W = data1.copy()
        target_attack(W, i)
        ssize += W.subgraph(max(nx.connected_components(W),
    key=len, default=[])).number_of_nodes()
        fremove += 1-float(W.number_of_nodes())/data1.
    number_of_nodes()
    results3.append((float(ssize)/n)/100)
    results4.append(float(fremove)/100)
print("r3:{}".format(results3))
print("r4:{}".format(results4))
fig4 = plt.figure(figsize=(16, 8), dpi=100)
plt.scatter(results3, results4, color = "black", marker = "o"
    )
plt.xlabel("f", fontsize=20)
plt.ylabel("S", fontsize=20)
plt.xlim(0, 1)
plt.ylim(0, 1)
```